Mangalore University

Network Anomaly Detection Models Using Machine Learning Algorithms

FINAL SYNOPSIS SUBMITTED

TO

MANGALORE UNIVERSITY

FOR THE AWARD OF THE DEGREE OF DOCTOR OF PHILOSOPHY

IN COMPUTER SCIENCE

BY

MOHAMMAD KAZIM

RESEARCH SCHOLAR

UNDER THE GUIDANCE OF

Dr. MANJAIAH D.H

PROFESSOR

DEPARTMENT OF POST-GRADUATE STUDIES AND RESEARCH IN COMPUTER SCIENCE,

MANGALORE UNIVERSITY,

MANGALAGANGOTHRI - 574 199.

September - 2022

Contents

1 Introduction and Literature Survey **3**
- 1.1 Overview of Cyber-security . . . 3
  - 1.1.i. Intrusion Detection Systems . . . 3
- 1.2 Problem Statement . . . 4
- 1.3 Datasets . . . 5
  - 1.3.i. UNSW-NB15 . . . 5
  - 1.3.ii. NSL-KDD . . . 5
- 1.4 Literature Review . . . 7
- 1.5 Objectives of the Proposed Research . . . 17

2 Feature Selection Using Ensemble Learning Approach for Network Traffic Data **18**
- 2.1 Introduction . . . 18
- 2.2 The Proposed Method . . . 18
- 2.3 Evaluation of the Proposed Method . . . 20
  - 2.3.i. Experimental Results . . . 21
- 2.4 Conclusion . . . 24

3 Network Anomaly Detection Using Ensemble Learning Approach **27**
- 3.1 Introduction . . . 27
- 3.2 The Proposed Method . . . 28
  - 3.2.i. Data Preprocessing . . . 28
  - 3.2.ii. Data Imbalance Processing . . . 30
- 3.3 Experimental Results and Analysis . . . 30
  - 3.3.i. Hyperparameter Optimization . . . 30
  - 3.3.ii. Classification . . . 32
- 3.4 Conclusion . . . 33

4 Deep Learning-based Network Anomaly Detection **35**
- 4.1 Introduction . . . 35
- 4.2 The Proposed Method . . . 36
  - 4.2.i. Data Preprocessing . . . 36
  - 4.2.ii. Data Imbalance Processing . . . 38
- 4.3 Experimental Results and Analysis . . . 38
  - 4.3.i. Classification . . . 39
  - 4.3.ii. Discussion . . . 40
- 4.4 Conclusion . . . 41

5 Protocol-wise Traffic Analysis for Network Anomaly Detection **43**
- 5.1 Introduction . . . 43
- 5.2 The Proposed Method . . . 44

5.2.i. Data Preprocessing . . . 45
5.2.ii. Data Division . . . 47
5.2.iii. Feature Selection . . . 48
5.2.iv. Sampling . . . 48
5.3 Experimental Results . . . 49
5.4 Results Analysis and Comparison . . . 54
5.4.i. Comparison . . . 54
5.4.ii. Discussion . . . 56
5.5 Conclusion . . . 57

6 Conclusion and Future Directions **58**
6.1 Conclusion . . . 58
6.2 Open Problems and Future Works . . . 59

List of Publications **60**

Abstract

Nowadays, cyber-security threats include a wide range, such as malware, DoS, phishing, social engineering, etc. An effective defence system can no longer rely solely on traditional security tools and techniques to counter these threats. Recently, Intrusion Detection Systems (IDS) are playing critical roles in network security. Intrusion detection systems are classified in several ways into sub-categories. One such classification is based on the detection method. IDS are mainly classified into two categories according to the detection method: misuses detection method or signature-based, and anomaly-based detection method. The signature-based detection method can detect only known attacks, and the signature database requires a continuous update. Anomaly-based detection methods can detect known and unknown (zero-day) attacks, but they generate a large number of false positives. Network Intrusion Detection Systems (NIDS) are a subset of IDS implemented on the network rather than on the host. Network Anomaly Detection Systems (NADS) are a subset of NIDS with the anomaly-based detection method. Despite many studies on NADS, it is still not implemented universally in real applications due to the challenges of a high false-positive rate.

Recently, machine learning algorithms and deep learning algorithms gained attraction in the purview of network anomaly detection systems. The datasets related to IDS are high dimensional and imbalanced. High dimensionality directly affects the accuracy and computational cost of the model's performance, the imbalance issue cause higher detection for the majority class and lower accuracy for the minority class. Our focus in this work is to design models using machine learning algorithms and deep learning methods to cope with the issues of network traffic data and classify anomalies and normal instances with higher accuracy and lower false-positive rate. Ensemble learning methods and deep learning techniques are widely used in network security. Ensemble learning enhances model performance for imbalanced data. Deep learning is employed due to the advantage of dealing with high dimensional data with non-linear and unbalanced properties and significant accuracy improvement.

In this research work four models are developed, i.e., a feature selection model, a network anomaly detection model using ensemble learning methods integrated with an imbalance processing technique, a network anomaly detection using deep learning techniques, and a network anomaly detection model using deep learning techniques on separate standard protocols. We evaluate the models on two commonly used datasets, i.e., UNSW-NB15 and NSL-KDD. Both datasets are freely available for research purposes. Feature selection is essential for prioritizing important attributes in data to improve prediction quality in machine learning algorithms. Different selection techniques identify different feature sets, so relying on a single method may result in risky decisions. We propose an ensemble learning-based feature selection method using union and quorum combination techniques with five primary individual selection methods. The individual methods are Analysis of Variance (ANOVA), Variance Threshold (VT), Sequential Backward Search (SBS), Recursive Feature Elimination (RFE), and Least Absolute Selection and Shrinkage Operator (LASSO). The proposed method reduces features in three rounds: i) discards redundant features using the pairwise correlation technique, ii) individual methods select their feature sets independently, and iii) equalizes individual feature sets. The equalized individual feature sets are combined using union and quorum techniques. The combined and individual sets are tested for network anomaly detection using Random Forest, Decision Tree, K-Nearest Neighbours, Gaussian Naive Bayes, and Logistic Regression classifiers.

Network anomaly detection systems are vital tools in network security. They are affected by data imbalance issues in classifying minority classes. Also, designing an efficient detection framework is sought to achieve a higher detection rate for minority classes, especially when utilizing ensemble learning methods. To overcome the issue of imbalanced data, we use a combined method of

oversampling and undersampling techniques. This imbalance processing tool integrates the Synthetic Minority Oversampling Technique and K-means clustering algorithm (SKM). Synthetic Minority Oversampling Technique (SMOTE) oversamples the minority class, and K-means is used to perform a cluster-based undersampling. We use Denoising Autoencoder (DAE) to select the top 15 features to reduce data dimensionality based on their higher weights. For anomaly detection, the XGBoost algorithm is deployed in the second model.

Artificial Neural Networks are pointing to promising results in the purview of network security. Convolutional neural networks (CNNs) are the specific architecture of feed-forward artificial neural networks (ANNs). It is the de-facto standard for various operations in machine learning and computer vision. To transform this performance toward the task of network anomaly detection, we use a one-dimensional convolutional neural network (1D-CNN) as a classification model in the third model of this research work. The denoising autoencoder (DAE) is used for feature selection to reduce data dimensions. To balance the data, we use a combined approach of oversampling technique, adaptive synthetic (ADASYN) and a cluster-based undersampling method using a clustering algorithm, K-means.

Finally, As the data distribution is different in different protocols in network traffic data, we propose a network anomaly detection model to treat the data based on separate protocols. Our approach embodied in the fourth model divides network traffic data into TCP, UDP, and OTHER protocol categories in the first phase, and then each category is treated independently. The majority classes are undersampled using the random undersampling technique. Feature selection is performed using the Chi-square technique, and then oversampling is conducted using the synthetic minority oversampling technique (SMOTE) to tackle the class imbalance problem in the dataset. We use a one dimensional CNN for classification.

Feature selection requires an intensive analysis, so automating it is a desire to speed up the detection process with minimum human intervention. Parameter optimization is a necessary task in many DL models; on the other hand, optimizing all the essential parameters requires lots of resources, so finding a way to minimize the resources required will be promising. Overlap problem in IDS data impacts model performance, especially in the case of multiclass classification. It requires some more work to find ways to treat overlapped classes appropriately.

Chapter 1

Introduction and Literature Survey

1.1 Overview of Cyber-security

The internet changed the manner of communication, which impacts different domains. In today's contemporary society, cyberspace plays a vital role in daily engagements such as education, healthcare, business, etc. The widespread use of digital appliances with various means of communication has increased humans' dependency on them to get applications and services anywhere at any time. On the other hand, with the constant growth of the internet and exponential growth of the number of digital devices in cyberspace, cyber attacks and threats are also getting more sophisticated day by day and creating various new vulnerabilities. Therefore, the significant role of cyber-security becomes a necessity. Cyber-security is a set of tools, practices and guidelines to protect network elements and data from attacks and unauthorized access. The goal is to avoid damages to the network and bring safety and privacy to network hardware, software and data [1]. Collaborative efforts of cyber-security professionals and researchers lead to designing a variety of cyber-defence systems for the purpose of maintaining confidentiality, integrity and availability of information in the cyber environment [2]. However, every day new cyber-security incident is happening, and security goals are flagrantly violated, but still, there is not an efficient framework in place to guarantee complete security. Figure 1.1 shows conventional cyber-security systems which address various threats of different forms at two levels, one is controlling at the network level, and another is controlling at the host level, which are called network-based and host-based defence systems, respectively [2].

1.1.i. Intrusion Detection Systems

A set of malicious activities attempting to exploit the principles of the Confidentiality, Integrity, and Availability (CIA triad) are called cyber threats [3]. The users initiating such threats are considered attackers. The attack's sophistication vector varies from one type to another, which is a big challenge for the detection process; for example, Packet Sniffing is a type of network attack where the attacker captures the data packets in travel and reads the sensitive data like passwords or card numbers and violate the confidentiality principle, malicious codes hijack program's flow execution, which violates the integrity principle, and DoS attacks break availability principle [4]. Various domains of cyber-security such as authentication, access control, cryptography, intrusion detection, perimeter security, risk and vulnerability assessment, forensics and auditing, and many more are in place to assure CIA triad principles for all components of computer network systems. Different security tools and mechanisms provide different protections; for example, vulnerability assessment is implemented to provide protection in the first

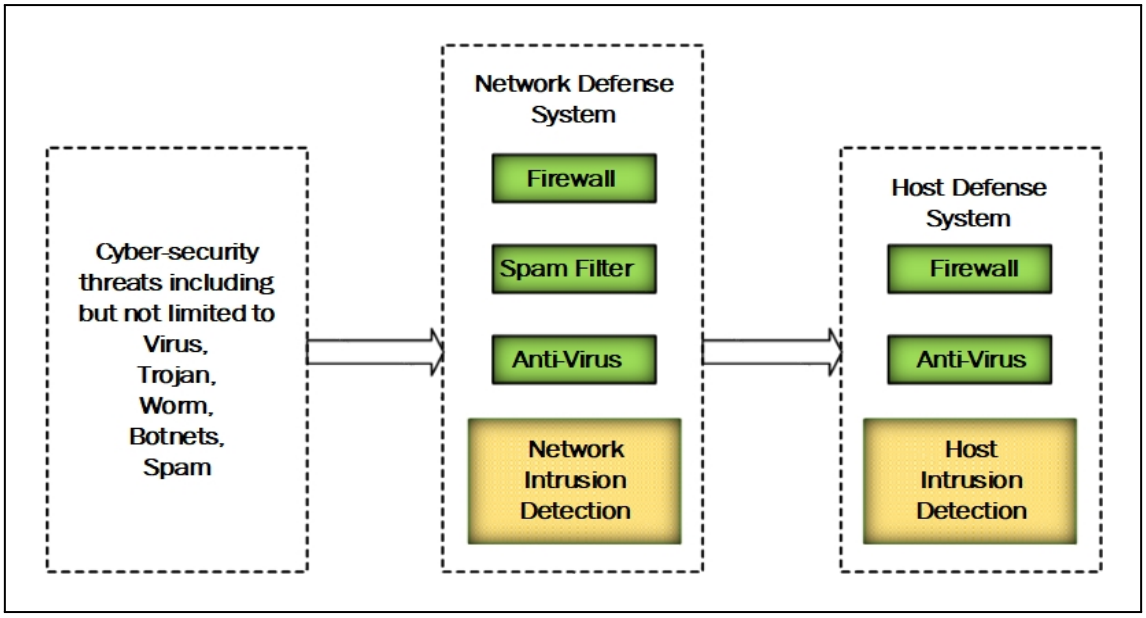


Figure 1.1: Conventional cyber-security system.

place, while anti-viruses and firewalls are deployed to defend against well-known attacks. Due to the rapid change in cyber-attack types and dimensions (fraud-related, cyber terrorism, cyber espionage, etc.), the traditional tools and mechanisms no longer suffice the needs of cyber defence systems. Intrusion Detection Systems (IDS) can be helpful in defining a defence in depth mechanism in order to monitor security breaches while the other prevention approaches are evaded. Intrusion Detection System was introduced by Denning in 1987 with the aim of security violation detection for a wider range. Intrusion Detection Systems can be hardware or software [5], and it can be classified in various ways [6–9], for example, classification based on the monitored environment can be host-based, network-based, and hybrid; classification based on detection methods can be misuse-detection (signature-based), anomaly-based detection, and stateful protocol analysis; classification based on deployment architecture can be distributed (compound system) and centralized (non-compound system); division based on the response of an IDS to an attack can be active and passive; and similarly based on decision making IDSs can be classified to co-operative and autonomous. In this research work, we focus on detection methods, especially anomaly-based techniques.

IDS can be classified mainly into two categories based on location in a network. One is Host-based IDS (HIDS), and the other is Network-based IDS (NIDS). Host-based IDS is installed in a host to monitor traffic coming in and going out from a particular host. NIDS is positioned exactly at the entry and exit point of data from the network to the outside world. Both HIDS and NIDS have their own advantages and disadvantages; before one decides which IDS suits his/her network environment the best s/he needs to have a clear concept of both types of IDSs.

1.2 Problem Statement

In this research work, we attempt to address the issue of developing effective models for identifying network traffic anomalies in large scale data. The rapid prediction of anomalous events with high accuracy and low false alarm rate from a large volume of network traffic data is one of the significant challenges for Network Anomaly Detection Systems (NADS).

This problem is split into two sub-problems:

- Utilizing the feature selection techniques to select the most relevant and optimal set of features is one of the significant challenges in designing NADS. Selecting compelling features from a dataset will decrease data dimensionality, ultimately decrease time complexity, and improve the model's performance.
- Designing effective models to differentiate between normal and anomalous traffic with high accuracy and low false alarm rate is another challenge for the NADS development process. Despite much research conducted in the domain, the methods suffer from a high ratio of false alarms in the detection process. Still, much work is required to tackle these issues. Developing models using ML and DL algorithms with effective variable selection and efficient imbalance processing seems promising.

1.3 Datasets

We use two commonly used datasets in this research work.

1.3.i. UNSW-NB15

UNSW-NB15 stands for University of New South Wales Network Based 2015 [10]. This dataset was generated in 2015 [11, 12], and is publicly available for research purposes. The dataset is available in four CSV files named as *UNSW-NB15_x.csv*, the x is numbered from 1 to 4. It has 2,540,044 samples in total, distributed in such a way that the first three files have 700,000 rows each, and the fourth file has 440,044 rows. There are nine types of attacks, such as Analysis, Worms, Shellcode, Reconnaissance, Exploits, Backdoor, Fuzzers, DoS, and Generic, in addition to Normal transactions. The total number of features is 49, including two labels, which are grouped into six categories: flow features, content features, basic features, time features, additional generated features and labelled features. additional generated features are further divided into two sub-categories called *general-purpose-features (from feature number 36 to 40)* and *connection-features (from feature number 41 to 47)*. Table 1.2 provides detailed information on each feature.

Table 1.1: NSL-KDD list of features.

No.	Feature Name	Feature Type	No.	Feature Name	Feature Type	No.	Feature Name	Feature Type
1	Duration	Numeric	15	Su Attempted	Binary	29	Same Srv Rate	Numeric
2	Protocol Type	Nominal	16	Num Root	Numeric	30	Diff Srv Rate	Numeric
3	Service	Nominal	17	Num File Creations	Numeric	31	Srv Diff Host Rate	Numeric
4	Flag	Nominal	18	Num Shells	Numeric	32	Dst Host Count	Numeric
5	Src Bytes	Numeric	19	Num Access Files	Numeric	33	Dst Host Srv Count	Numeric
6	Dst Bytes	Numeric	20	Num Outbound Cmds	Numeric	34	Dst Host Same Srv Rate	Numeric
7	Land	Binary	21	Is Hot Logins	Binary	35	Dst Host Diff Srv Rate	Numeric
8	Wrong Fragment	Numeric	22	Is Guest Login	Binary	36	Dst Host Same Src Port Rate	Numeric
9	Urgent	Numeric	23	Count	Numeric	37	Dst Host Srv Diff Host Rate	Numeric
10	Hot	Numeric	24	Srv Count	Numeric	38	Dst Host Serror Rate	Numeric
11	Num Failed Logins	Numeric	25	Serror Rate	Numeric	39	Dst Host Srv Serror Rate	Numeric
12	Logged In	Binary	26	Srv Serror Rate	Numeric	40	Dst Host Rerror Rate	Numeric
13	Num Compromised	Numeric	27	Rerror Rate	Numeric	41	Dst Host Srv Rerror Rate	Numeric
14	Root Shell	Binary	28	Srv Rerror Rate	Numeric	42	Label	Nominal

1.3.ii. NSL-KDD

NSL-KDD stands for Network Security Laboratory-Knowledge Discovery in Database [10]. The KDDCup99 dataset has redundant and duplicated records, which causes a poor evaluation for

Table 1.2: UNSW-NB15 list of features.

No.	Feature Name	Feature Type	Description
		Flow Features	
1	Srcip	Nominal	IP Address of Src.
2	Sport	Integer	Port Number of Src.
3	Dstip	Nominal	IP Address of Dest.
4	Dsport	Integer	Port Number of Dest.
5	Proto	Nominal	Transaction Protocol.
		Basic Features	
6	State	Nominal	State of Dependent Protocol, like 'ACC', 'CLO', 'CON', else(-).
7	Dur	Float	Total Duration of The Row.
8	Sbytes	Integer	Src. to Dest. Transaction Bytes.
9	Dbytes	Integer	Dest. to Src. Transaction Bytes.
10	Sttl	Integer	Src. to Dest. 'TTL' Value.
11	Dttl	Integer	Dest. to Src. 'TTL' Value.
12	Sloss	Integer	Src. Packets Re-transmitted or Dropped.
13	Dloss	Integer	Dest. Packets Re-transmitted or Dropped.
14	Service	Nominal	like 'HTTP', 'SSH', 'FTP', 'DNS', 'SMTP', 'FTP-Data' ,'IRC' , Else(-).
15	Sload	Float	Src. bits Per Sec.
16	Dload	Float	Dest. bits Per Sec.
17	Spkts	Integer	Src. to Dest. Packet Count.
18	Dpkts	Integer	Dest. to Src. Packet Count.
		Content Features	
19	Swin	Integer	Value of Src. 'TCP' Window Advertisement.
20	Dwin	Integer	Value of Dest. 'TCP' Window Advertisement.
21	Stcpb	Integer	Src. 'TCP' Base Sequence-Number.
22	Dtcpb	Integer	Dest. 'TCP' Base Sequence-Number.
23	Smeansz	Integer	Mean of Flow Packet-Size Transmitted by Src.
24	Dmeansz	Integer	Mean of Flow Packet-Size Transmitted by Dest.
25	Trans_depth	Integer	Pipelined Depth Into Connection of 'HTTP' Request/Response Transaction.
26	Res_bdy_len	Integer	Actual Uncompressed Content Size of The Data Transferred From The Server's 'HTTP' Service.
		Time Features	
27	Sjit	Float	Src. jitter (m-sec).
28	Djit	Float	Dest. jitter (m-sec).
29	Stime	Timestamp	Row Start-Time.
30	Ltime	Timestamp	Row last-Time.
31	Sintpkt	Float	Src. Inter-Packet Arrival-Time (m-sec).
32	Dintpkt	Float	Dest. Inter-Packet Arrival-Time (m-sec).
33	Tcprtt	Float	Round-Trip Time of 'TCP' Conn. Setup , Calculated by The Sum of 'ackdat' And 'synack'.
34	Synack	Float	Conn. Setup Time of 'TCP' Between 'SYN' And The 'SYN_ACK' Packets.
35	Ackdat	Float	Conn. Setup Time of 'TCP' Between 'SYN_ACK' And 'ACK' Packets.
		Additional Generated Features (36-40 are general purpose features and 41-47 are connection features)	
36	Is_sm_ips_ports	Binary	If srcip = dstip And sport = dsport, value is 1, 0 Otherwise.
37	Ct_state_ttl	Integer	No. of Each State (6) According to Specific Values For Src's/Dest's TTL(10)(11).
38	Ct_flw_http_mthd	Integer	No. of Flows Obtained Using Methods like Get And Post In 'HTTP' Service.
39	Is_ftp_login	Binary	Value Is 1 If 'FTP' Session Is Accessed by User's Credential, 0 Otherwise.
40	Ct_ftp_cmd	Integer	No. of Flows Using A Command In 'FTP' Session.
41	Ct_srv_src	Integer	No. of Connections Having Same service (14) And srcip (1) within 100 Transaction.
42	Ct_srv_dst	Integer	No. of Connections Having Same service (14) And dstip (3) within 100 Transaction.
43	Ct_dst_ltm	Integer	No. of Connections Having Same dstip (3) within 100 Transaction.
44	Ct_src_ ltm	Integer	No. of Connections Having Same srcip (1) within 100 Transaction.
45	Ct_src_dport_ltm	Integer	No of Connections Having Same srcip (1) And The dsport (4) within 100 Transaction.
46	Ct_dst_sport_ltm	Integer	No of Connections Having Same dstip (3) And The sport (2) within 100 Transaction.
47	Ct_dst_src_ltm	Integer	No of Connections Having Same srcip (1) And The dstip (3) within 100 Transaction.
		Labelled Features	
48	Attack_cat	Nominal	The Name of Attack Categories, i.e., Analysis, DoS, Exploits, Backdoor, Generic, Fuzzers, Reconnaissance, Shellcode, and Worms.
49	Label	Binary	1 For Attack And 0 For Normal Records.

anomaly detection in NADSs. The issue has been resolved in the refined version of it, which is the NSL-KDD dataset. NSL-KDD have different files with different formats [13, 14]. We considered KDDTrain+.TXT, which has 125,973 data points and KDDTest+.TXT, which has 22,544 data points. It has 41 features in total and a class label. The features' types are numeric, binary and nominal. There are four major categories of attacks in the dataset: Probe, DoS, R2L, U2R, and Normal class. Further details on the dataset is given in Table 1.1.

1.4 Literature Review

In this section, we review the most related work to our research. We have considered four main aspects in our literature review: feature selection, ensemble learning, imbalance processing, and DL methods for network anomaly detection.
In work [15], the authors proposed a feature selection method, Random Forest (RF) forward selection ranking and RF backward elimination ranking. In the first approach, the top three features with the highest weights are kept in the first subset, and all the other features are put separately in the second subset; then, features from the second subset are added one by one into the first subset to check the detection rate. In the second approach, all the features are kept in one set, and one feature with the lowest weight is eliminated at a time to check the impact on the detection rate.
The authors in [16] proposed a clustered mutual information hybrid feature selection approach. The variable clustering was conducted in unsupervised steps based on similarity, and supervised learning was then applied to select optimal features to boost similarity with response features representing class labels. In work, [17], an analysis of network traffic variables for anomaly detection is rendered. The authors proposed a feature selection method using wrappers and filters techniques. The method relies on variable ranking and weighting to identify a temporary baseline of features with higher importance. In the next step, the selected set of features is passed through a refinement process applying brute force search analysis to reduce the number of features. The refinement process is composed of two steps: the first one is starting with all number of features and eliminating features one by one (backward elimination) with the stop criterion of maximum performance, and the second one is taking the baseline features and adding new features (forward selection) until the improvement accuracy falls below the threshold. The proposed method imposes much computational cost due to the nature of brute force implemented.
The authors in [18] proposed a features selection method using the feature ranking concept using two algorithms which are information gain and correlation. They used a combination of both algorithms to reduce the number of features for network intrusion detection. Their assumption was that the features with higher entropy values contain more information, while the features with lower correlation scores would be more helpful in differentiating classes in multiclass classification problems. The authors claimed over 94% Accuracy using neural network classifier with the reduced number of features (25 out of 41 features) on KDDCup99. In reference [19], the authors used two feature reduction algorithms, Canonical Correlation Analysis (CCA) and Linear Discriminant Analysis (LDA). They evaluated their methods using seven classifiers (Rep Tree, Naive Bayes, RF, Random Tree, Random Committee, Bagging Randomizable and Filtered). The performance of 7 different classifiers was evaluated based on various parameters such as Accuracy, Specificity, Kappa, Mean Absolute Error (MAE), FPR, Precision, Recall, ROC area, and training time. It was found that the LDA feature selection method with a random tree classifier was an outperformer. They also highlighted that the results with the UNSW-NB15 dataset were more realistic compared to the results with the same purpose using KDDCup99 and its variant NSL-KDD. In reference [20], the authors investigated feature selection against IDS-2017. In the first step, each feature's contribution is obtained with the help of a rule-based classification tool (JRip), and later 18 top scorer features are chosen as the optimal subset. The

selected features are evaluated for classification using K-Nearest Neighbours (KNN) and C4.5 Decision Tree (DT) algorithms. However, the Accuracy is reported 99%, but the study considers only DDoS attacks.

Ensemble learning methods play a vital role in the domain of network anomaly detection [21, 22]. Authors in [23] designed an ensemble-based intrusion detection system to identify network intrusions, especially those that attempt to exploit well-known IoT protocols. The authors in [24] implemented different classification methods such as Naive Bayes (NB), KNN, SVM, Fuzzy C-Means, Radial Basis Function, and ensemble technique for IDS. In the work [25], the authors proposed an adaptive ensemble learning-based IDS in order to detect different attack types such as Denial of Service (DoS), R2L, U2R, and Probe. To improve detection accuracy, they used a combination of multiple base classifiers such as DT, KNN, RF, and Deep Neural Network (DNN). Liu et al. [26] introduced a possible world-based AdaBoost (PwAdaBoost) algorithm to cope with the issue of uncertain data categorization. The authors claim that this method is capable of handling different uncertain data, so it can be employed to deal with uncertain data in IDS. Authors in [27] proposed an ensemble method using three classifiers, each focusing on the majority, middle, or tail groups of classes. Authors in [22] conducted a survey on hybrid learning approaches as well as on ensemble learning-based methods. The two main multiclass classifier systems, homogeneous ensembles (base classifiers with the same classification paradigm) and heterogeneous ensembles (base classifiers with different classification paradigms), were highlighted. They concluded that heterogeneous with weighted majority voting is rarely deployed for IDSs. Tang et al. [28] designed an improved version of the AdaBoost algorithm (MF-Adaboost) to detect lower-rate denial of service (LDoS) attacks in communication networks. The method identifies some of the features which are more informative in classifying classes.

According to [29, 30], DL algorithms produce higher accuracy and lower false alarms compared to traditional ML algorithms. In traditional MLs, the computational task is large, and feature extraction efficiency is poor. Deep Learning is capable of representing low-level intrusion traffic data features as high level and can capture correlations and features automatically in a large data [30]. It has demonstrated success in various domains of real-world problems. Hence, they have been increasingly deployed for IDSs in recent years. The authors in [31] conducted a comparative analysis on different dimensionality reduction techniques such as Principal Component Analysis PCA), t-Distributed Stochastic Neighbour Embedding (t-SNE), Isomap, and Interpol. They used RF for classification purposes. The Isomap with RF outperforms for small dimension data. The survey conducted in [32] focused on different datasets used in NIDS, and the performance of seven DL models was evaluated on two new datasets. In [33], the authors used an unsupervised method to minimize human involvement in IDS and also boost the detection accuracy. A stacked non-symmetric autoencoder was the method used in this work. In [34], the authors presented a survey on DL algorithms used in anomaly-based detection systems; they also considered the taxonomy of IDSs and future research directions. The work [35] evaluated the effectiveness of various DL models on KDDCup99 dataset. The authors used a Convolutional Neural Network (CNN) and some combination of CNN with LSTM, a Recurrent Neural Network (RNN) and a gated recurrent unit. The results show that the simple CNN outperforms in binary classification, whereas the CNN-LSTM surpasses other architectures in multiclass classification. The authors in [36], used RNN and CNN for NSL-KDD dataset classification. They focused on solving data imbalance issues by using a cost function method. Based on the training sample number of each class, the cost function weight coefficient is set. The result shows that DL models were performing better than the traditional algorithms such as NB, NB tree, RF, J48, and SVM. Although DL has shown great success in various applications, its performance is not ideal when the data is imbalanced [10].

Class imbalance issues have attracted attention in recent years; it refers to the imbalance of classes in the data; for example, there are many instances in one class while very few instances in another class [37]. According to [38], the current solutions for this challenge are broadly

divided into four types: cost-sensitive methods, integration methods, algorithm level methods, and data-level methods. The data level methods are commonly used due to their independence from algorithms and using simple operations. The main idea behind that is resampling, including undersampling and oversampling. Some of the other representative methods are ADASYN [39], SMOTE [40], BalanceCascade [41], and EasyEnsemble [41]. Oversampling has shown the best result while dealing with imbalanced data in DL [42], the advantage is that it does not cause overfitting. However, it causes some increases in the data volume, which ultimately increases computational cost. To overcome this challenge, a combination of oversampling and undersampling techniques is recommended. Authors in [43] proposed a Cluster-SMOTE sampling technique for their model that fuses undersampling and oversampling. Authors in [44] proposed SMOTE oversampling technique with an adaptive boosting ensemble learning-based network anomaly detection in IoT network. In [45], the authors claimed that the combination of the fusion of clustering undersampling and AdaBoost had shown a better anomaly detection performance in network traffic. Abhishek Divekar et al. [46] used SVM, RF, NB, K-means, Neural Network, and DT algorithms for the classification purpose of KDDCup99 and NSL-KDD datasets. They also applied SMOTE oversampling and random undersampling techniques to create a balanced version of datasets, and they found that skewed target classes in NSL-KDD and KDD-99 hamper the efficacy of classifiers on minority classes (R2L and U2R), which is leading to possible security risks. They also explored the performance of the classifiers on the UNSW-NB15 dataset before and after SMOTE for the minority classes and highlighted that the trained classifiers on UNSW-NB15 yield higher F-score compared to those trained on KDDCup99 and NSL-KDD datasets in binary classification.

Table 1.3: Summary of Literature Review.

Begin of Table							
No.	**Year**	**Authors**	**Title**	**Applications**	**Algorithms**	**Model Performance**	**Dataset**
1	2020	Sharma et al. [27]	Long-tailed recognition using class-balanced experts	Imbalance handling ensemble method	KL-div min, Soft-Voting, Expert Selection, Stacking, Joint Calibration	NA	ImageNet-LT, Places-LT
2	2020	Tang et al. [28]	MF-Adaboost: LDoS attack detection based on multi-features and improved Adaboost	Feature selection, imbalance processing, and LDoS attack detection method	Feature selection: Chi-squared test algorithm Imbalance processing: sample weight adjustment algorithm Classification: improved AdaBoost (MF-AdaBoost)	MF-AdaBoost DR on NS2 = 94.05% MF-AdaBoost DR on Test-bed =97.06%	NS2 simulation platform real test-bed environment
3	2020	Ferrag et al. [32]	DL for cyber security intrusion detection: Approaches, datasets, and comparative study	Review of 35 well-known cyber datasets and analysing 7 DL algorithms on CSE-CIC-IDS2018 and Bot-IoT datasets	RNN, DNN, RBM, DBN, CNN, DBMs, DA	According to this survey some of the commonly used datasets are: DARPA, NSL-KDD, UNSW-NB15, ISCX dataset, CTU-13, TWENTE	CSE-CIC-IDS2018, Bot-IoT
4	2020	Aldweesh et al. [34]	DL approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues	This survey reviewed and compared the key surveys considering DL for intrusion detection and built the current survey upon the previous ones.	AE, DBN, RNN, CNN	Earlier DL architecture used in IDS are AE, DBN, RNN, while research works on CNN recently appeared in 2017. The majority of proposed IDS used KDDCup99 and NSL-KDD	KDDCup99, NSL-KDD etc.

Continuation of Table 1.3

No.	year	Authors	Title	Applications	Algorithms	Model Performance	Dataset
5	2020	Dash et al. [44]	Model based IoT security framework using multiclass adaptive boosting with SMOTE	Imbalance processing and network anomaly detection	LR_SMOTE, MLP_SMOTE, NB_SMOTE, LDA_SMOTE, DT_SMOTE, RF_SMOTE, Proposed AdaBoost_SMOTE	Proposed AdaBoost_SMOTE is found superior over the other methods in terms of Accuracy, Precision, and F-score	IoT network data DS2OS
6	2020	Huan et al. [45]	Anomaly detection method based on clustering undersampling and ensemble learning	Imbalance processing and network anomaly detection	Cluster based undersampling + AdaBoost	DR. on Moore = 72.4% DR. on iscxvpn2016 = 82.1%	Moore, iscxvpn2016
7	2019	Gao et al. [25]	An adaptive ensemble ML model for intrusion detection	Ensemble intrusion detection	SVM, KNN, LR, AdaBoost, RF, NN	Adaptive ensemble method Acc. = 85.2%	NSL-KDD
8	2019	Alhakami et al. [47]	Network anomaly intrusion detection using a nonparametric Bayesian approach and feature selection	Feature selection and anomaly detecion	Proposes a Bayesian-based classifier for network anomaly detection. A feature selection mechanism has been integrated into the proposed method	The best values for infinite models is found for the infinite bounded generalized Gaussian InBGG with 83.49%, 87.41%, and 90.40% for KDDCup99, Kyoto 2006+ and ISCX datasets, respectively.	KDDCup99, Kyoto 2006+, ISCX
9	2018	Dahiya et al. [19]	Network Intrusion Detection in Big Dataset Using Spark	Feature selection and intrusion detection	Feature selection: CCA & LDA & Classification: Rep Tree, NB, RF, Random Tree, Random Committee, Bagging Randomizable and Filtered	LDA and Random Tree are the winners	UNSW-NB15, NSL-KDD, KDDCup99

Continuation of Table 1.3

No.	year	Authors	Title	Applications	Algorithms	Model Performance	Dataset
10	2018	Patil et al. [20]	Feature reduction in flow based intrusion detection system	Feature selection and intrusion detection	Feature selection: JRip ML algorithm Classification: IBk and PART	IBk Acc. =99.96%, PART Acc. =99.94% (only for DDoS LOIC attack)	CICIDS2017
11	2018	Moustafa et al. [23]	An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things	Ensemble intrusion detection	AdaBoost ensemble learning method is developed using DT, NB, and ANN	The method shows that DR and FPR are approximately between 95.25% and 99.86% and between 0.01% and 0.72%, respectively.	UNSW-NB15, NIMS
12	2018	Zaman et al. [24]	Evaluation of machine learning techniques for network intrusion detection	Ensemble intrusion detection	KM, KNN, FCM, NB, SVM, RBF, and ensemble of all	KM Acc. = 83.6% KNN Acc. = 97.54% FCM Acc. = 83.6% NB Acc. = 96.72% SVM Acc. = 94.26% RBF Acc. = 97.54% Ensemble Acc. = 96.72%	Kyoto 2006+
13	2018	Berger et al. [29]	Comparative study of machine learning methods for in-vehicle intrusion detection	Comparative analysis of MLs and DLs for anomaly detection	SVM,NN, LSTM	LSTMs present a practical and potentially the most powerful approach of anomaly detection out of the methods analyzed in this work	CAN bus data
14	2018	Shone et al. [33]	A DL approach to network intrusion detection	Intrusion detection	RF+ new method called non-symmetric deep autoencoder (NDAE)	RF+NDAE Avg. Acc. = 97.85% using KDDCup99 RF+NDAE Avg. Acc. = 85.42%% using NSL-KDD	KDDCup99, NSL-KDD

Continuation of Table 1.3

No.	year	Authors	Title	Applications	Algorithms	Model Performance	Dataset
15	2018	Wu et al. [36]	A novel intrusion detection model for a massive network using convolutional neural networks	Imbalance processing and Intrusion detection	NB, NB tree, RF, J48, and SVM, RNN, CNN	CNN and RNN perform higher than traditional algorithms. However, CNN scores lower accuracy than RNN, but the FAR for CNN is lower than RNN	NSL-KDD
16	2018	Buda et al. [42]	A systematic study of the class imbalance problem in convolutional neural networks	Imbalance impact on CNN	CNN	oversampling does not cause overfitting of convolutional neural networks, as opposed to some classical machine learning models	MNIST, CIFAR-10, ImageNet
17	2018	Divekar et al. [46]	Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD-Cup99 alternatives	Imbalance processing and network anomaly detection	NN, SVM, DT, RF, NB, K-Means	UNSW-NB15 is a modern substitute to KDDCup99 and NSL-KDD	KDDCup99, NSL-KDD, UNSW-NB15
18	2017	Manzoor et al. [18]	A feature reduced intrusion detection system using ANN classifier	Feature selection	Information gain and correlation	NN Acc. =94%	KDDCup99
19	2017	Vanerio et al. [21]	Ensemble-learning approaches for network security and anomaly detection	Intrusion detection	Base learners: DT, NB, Neural Net, SVM, KNN, RF, logreg, CART. Weights: Mvaccuracy, Mvexp, MVuniforme	MVexp Super Learner performs the best for all attack and anomaly types in both datasets	MAWILab and semi-synthetic
20	2017	Aburomman et al. [22]	A survey of intrusion detection systems based on ensemble and hybrid classifiers	Survey on ensemble and hybrid classification techniques	Reviewed approaches: Stacking, Averaging, Weighted averaging, Belief measurement, Dempster-Schafer combination, Majority voting, and Weighted majority voting	Voting-based is a common way of combining classifiers	NSL-KDD, KDDCup99

Continuation of Table 1.3

No.	year	Authors	Title	Applications	Algorithms	Model Performance	Dataset
21	2017	Yin et al. [30]	A DL approach for intrusion detection using recurrent neural networks	Intrusion detection	J48, ANN, RF, SVM, RNN	RNN DR. = 97.09%	NSL-KDD
22	2017	Vinayakumar et al. [35]	Applying convolutional neural network for network intrusion detection	Intrusion detection	MLP, CNN, CNN-RNN, CNN-LSTM and CNN-GRU	CNN outperforms in binary classification, whereas the CNN-LSTM surpasses other architectures in multiclass classification	KDDCup99
23	2016	Yuan et al. [48]	A novel semi-supervised adaboost technique for network anomaly detection	Anomaly detecion	Integrates ensemble and semi-supervised learning technique to create a new AdaBoost algorithm for network anomaly detection purposes.	Proposed method Acc. = 96.63% DR. = 88.58%	KDDCup99, NSL-KDD
24	2015	Iglesias et al. [17]	Analysis of network traffic features for anomaly detection	Feature selection	Filters and stepwise regression wrappers	DTC Acc. = 78.22%, kNN Acc. = 76.65%, Bayes Acc. = 73.38%, ANN Acc. = 79.25%, SVM Acc. = 78.83%	NSL-KDD
25	2014	Al-Jarrah et al. [15]	Machine-learning-based feature selection techniques for large-scale network intrusion detection	Feature selection	RF forward selection ranking and RF backward elimination ranking	RF DR=99.8%, RF FPR=0.001%	KDDCup99, NSL-KDD

Continuation of Table 1.3

No.	year	Authors	Title	Applications	Algorithms	Model Performance	Dataset
26	2014	Liu et al. [16]	A mutual information-based hybrid feature selection method for software cost estimation using feature clustering	Feature selection	Wrapper method	The proposed method performs better over some supervised and unsupervised feature selection methods	ISBSG R8, Desharnais
27	2014	Heider et al. [31]	Unsupervised dimension reduction methods for protein sequence classification	Dimensionality reduction and classification	Dimensionality reduction: PCA, Isomap, Interpol, t-SNE Classification: RF	Interpol + RF performs better	Two artificial and eighteen real-world protein datasets
28	2011	Galar et al. [38]	A review on ensembles for the class imbalance problem: bagging, boosting-, and hybrid-based approaches	Imbalance processing	SMOTE and its combination	The solutions for imbalance issue are broadly divided into four types: cost-sensitive methods, integration methods, algorithm level methods, and data-level methods. The data level methods are commonly used due to their independence from algorithms and using simple operations	Used a set of 44 two-class real-world problems, which suffer from the class imbalance problem
29	2008	He et al. [39]	ADASYN: Adaptive synthetic sampling approach for imbalanced learning	Imbalance processing	Proposed adaptive synthetic (ADASYN)	ADASYN can adaptively generate synthetic data samples for the minority class to reduce the bias introduced by the imbalanced data distribution	Vehicle, Diabetes (PID), Vowel, Ionosphere, Abalone
30	2008	Liu et al. [41]	Exploratory undersampling for class-imbalance learning	Imbalance processing	Propose EasyEnsemble and BalanceCascade	Both EasyEnsemble and BalanceCascade are efficient	16 UCI

Continuation of Table 1.3

No.	year	Authors	Title	Applications	Algorithms	Model Performance	Dataset
31	2002	Chawla et al. [40]	SMOTE: synthetic minority oversampling technique	Imbalance processing	Proposed SMOTE	SMOTE approach can improve the accuracy of classifiers for a minority class	Pima, Phoneme, Adult, E-state, Satimage, Forest Cover, Oil, Mammography, Can

End of Table

1.5 Objectives of the Proposed Research

The objectives of the proposed research work on network anomaly detection models using machine learning algorithms are:

- Designing a machine learning-based model to select effective features from network traffic data to reduce time complexity.
- Utilizing ensemble learning methods using machine learning algorithms to improve anomaly detection in network traffic data.
- Proposing a deep learning-based model for network anomaly detection.
- Analyzing network traffic data based on separate standard protocols for network anomaly detection.

Chapter 2

Feature Selection Using Ensemble Learning Approach for Network Traffic Data [1]

2.1 Introduction

Recently, high dimensionality and a large volume of data have emerged in various ML applications. Referring to [49], the high-dimensionality issue not only incurs high computational cost and unbearable resource requirements but also deteriorates the generalisation capability of the model. With the emergence of the fast-growing number of high-dimensional datasets, feature selection also gained more interest in the field of ML. Feature selection is a technique that detects relatively features with higher importance and ignores the less important ones. The goal is to obtain a superior feature set that can optimise the learning complexity of a model and describe the given problem properly with minimum performance degradation. On the other hand, dimensionality reduction tools, like PCA and t-SNE, provide a projection of given features into a lower number of dimensions, but the main difference is that they are incapable of preserving individual feature definitions while feature selection techniques do [50, 51]. In this chapter, we propose a novel feature selection method using an ensemble learning approach, employing some of the commonly used selection techniques from various feature selection classes. As the selection methods are independently executed, so each one identifies a different feature set. To see the potential power of incorporation of different methods, we combine the results of individual methods using union and quorum based on set theory and compare anomaly detection performance of each combined feature set and individual sets using RF, DT, KNN, Gaussian Naive Bayes (GNB), and LR classifiers with publicly available network traffic datasets UNSW-NB15 and NSL-KDD.

2.2 The Proposed Method

The proposed method utilizes a group of feature selection methods from different classes (filter, wrapper, embedded) to identify the most relevant features that potentially can be better than others for anomaly detection. Figure 2.1 demonstrates our proposed feature selection method. The model is composed of two independent rounds, with multiple feature selection techniques and an equalizer round at the final stage. The first round discards highly redundant attributes based

[1] Doreswamy, Mohammad Kazim Hooshmand, and Ibrahim Gad. **Feature Selection Approach Using Ensemble Learning for Network Anomaly Detection**. *CAAI Transactions on Intelligence Technology.* 2020, Vol. 5, Issue 4, pp. 283–293, ISSN: 2468-2322.

on pair-wise correlation factors and reduces the number of variables from N to M. The second round employs uni-variate methods (VT and ANOVA), SBS, RFE, and LASSO to identify their own set from M set of features. Each technique reduces the number of features independently and identifies different sets with different numbers, given as g, h, i, j, and k. Parameters setting

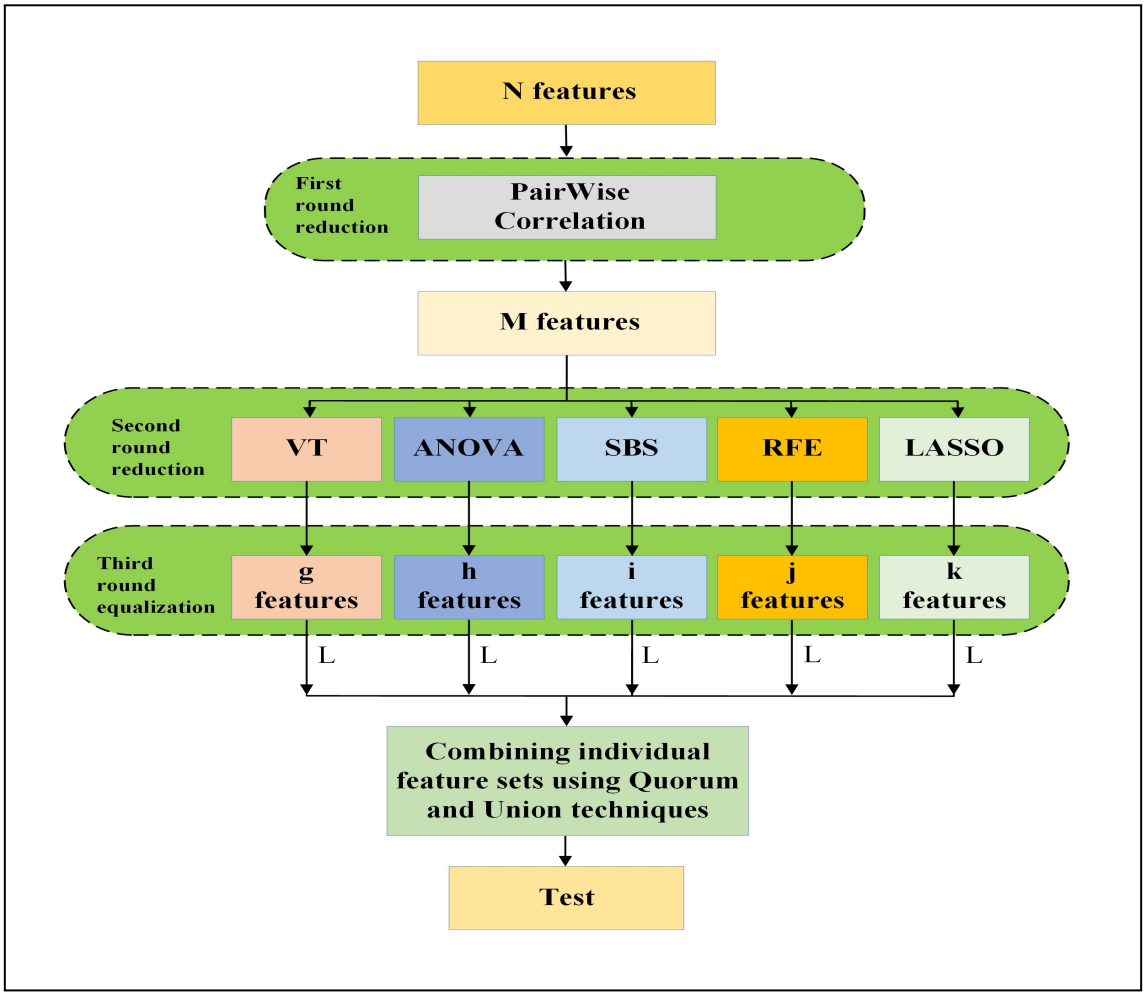


Figure 2.1: The proposed feature selection model.

and configuration of feature selection methods are discussed in section 2.3. The third round takes the only common number of features (L) from each feature set, identified by individual methods; in this case, the common number for UNSW-NB15 is L=12 and for NSL-KDD L=20. The values 12 and 20 came from the minimum number of features identified by individual methods, and the necessity of equalization is to make a comparison in the same span of number. This value may change dynamically based on the sample size of data. The model automatically manages all three rounds except for the parameters setting of selection methods. To examine selection methods from different feature selection classes, we tested Cross-validation(CV) on the training set and the anomaly detection performance on the test set. Figure 2.2(a) shows the cross-validation result for each 12 iterations of selection methods. We observe that the ANOVA subset shows a small degradation around six features and a significant degradation around three features. Variance Threshold subset shows a small degradation around eight features and a significant degradation around three features. On the other hand, SBS, RFE and LASSO behave very similar and show a significant drop around six features and a bit up and down between 2 and 6 features. The CV was examined on the train set using RF, and the proportion of train-test will be 70% and 30% through this study. We analyze the anomaly detection performance using

a test set to see the goodness of selected feature sets. Figure 2.2(b) shows the actual testing result of all subsets using the RF classifier. The results show that the accuracy trend is almost similar to the one with a training CV, which is good. Then we selected the SBS subset as an example to see how the same subset of features performs with different classifiers. The test result is given in Figure 2.2(c), showing that SBS results fluctuate for different classifiers. For example, results around six features are similar for RF, DT, and KNN but significantly low for LR and GNB. Also, we tested our method on the NSL-KDD dataset; the result is given in Figure 2.3. Figure 2.3(a) shows the cross-validation result, Figure 2.3(b) shows the actual testing result of all subsets using RF classifier and Figure 2.3(c) provides result of different classifiers with SBS subset. The overall behaviour of the method is almost similar in both datasets. So we can claim that certain selection techniques may perform better with certain classifiers and vice-versa, so choosing a single selection technique without comparative analysis may lead to a risky decision. Our proposed method combines different selection techniques to minimize the possible risk. We use union and quorum techniques to combine the individual results of different selection methods and suggest new sets. Quorum takes the variables, which are selected by the majority of individual techniques; in this case, the threshold is greater than and equal to 3 because we have five selectors. And the union takes those variables which are suggested by any of the methods.

2.3 Evaluation of the Proposed Method

The evaluation of the proposed method and experimental results of this study are presented in this section. Preprocessing and normalization of data is done before the data is being used by classifier. Most parameters of feature selectors in the second round of our proposed method were set to default, except some of them, which are manually configured, and the details are provided in Table 2.1. For feature selectors, we tested different values of parameters manually with the goal of keeping the number of features between 10 to 20 for individual methods just to stand on a fair number of features. In the case of the classifies, we have not done any optimization for parameter setting, as our goal is to simply compare different results of feature sets with unified parameters.

Table 2.1: Feature selectors/classifiers parameter setting.

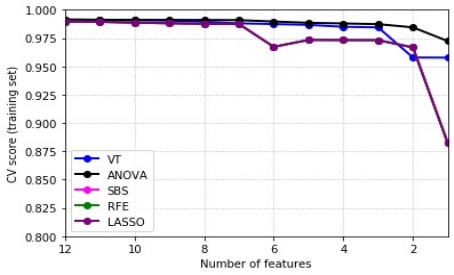
Method	**Parameters**
VT	threshold=0.005
ANOVA	f_classif, percentile=70
SBS	RandomForestClassifier(n_jobs=-1 ,n_estimators=5), k_features=20, forward=False, floating=False, verbose=1, scoring='roc_auc', cv=3
RFE	estimator=RandomForestClassifier(n_jobs=-1, n_estimators=5), n_features_to_select=20, step=1
LASSO	LogisticRegression(C=1, penalty='l2')
RF	n_estimators=100
DT	criterion = 'entropy', random_state = 0
KNN	n_neighbors = 5, metric = 'minkowski', p = 2
GNB	default
LR	solver='sag'

2.3.i. Experimental Results

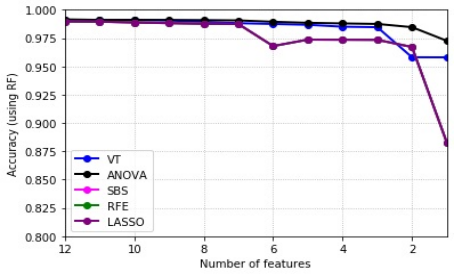
Referring to Figure 2.1, features identified by individual selection methods are equalized in round three and then incorporated using quorum and union. The number of selected features using the UNSW-NB15 is 14 and 25 from quorum and union techniques, respectively. Similarly, using the NSL-KDD, the number of selected features are 19 and 36 from quorum and union, respectively. We calculate F-score for RF, DT, KNN, GNB and LR with both the quorum and union set iteratively to see the optimal set of features. The results are shown in Figure 2.2(d) and Figure 2.2(e).
We compare union and quorum features in terms of F-score to see which one is performing better with a lesser number of features. From Figures 2.2(f)-2.2(j) we can see that quorum set is performing better with all classifiers with very less number of features, and similar behavior is seen in Figures 2.3(f)-2.3(j).

From Figure 2.2(d), we observe that RF performance was the highest with the quorum feature set. It scored 0.99 for 12 features and gained 0.98 for 2-11 features. The second and third top performers were DT and KNN. GNB shows a significant drop around nine features, and LR has a drastic drop around seven features. From Figure 2.2(e), we observe that again, RF is leading the performance with the use of the union feature set, as it scored 0.99 between 6-25 features with small variation and showed a significant drop around three features. The second and third good performers are DT and KNN. Decision Tree and KNN scored 0.99 between 10-25 features and 0.98 between 3-9 features. Despite fluctuations, GNB scores surprisingly 0.98 around 12 features, which is the highest through all the iterations, and a significant drop is noticed around ten features. Likewise, LR scores 0.98 between 19-25 features and 0.97 between 16-18 features, which is boosted between 11-15 features to 0.98 again. LR shows a drastic drop around nine features. From these two figures, we observe that RF is the winner with both the quorum and union sets. It is F-score value is 0.99 with six features with the union set and 12 features with the quorum set. We analyzed the performance of each classifier independently with both the quorum and the union feature sets. Figures 2.2(f)-2.2(j) provide a comparison of individual classifier performance with both the incorporated feature sets with respect to a common span of iterations. From Figure 2.2(f), Figure 2.2(g) and Figure 2.2(h), we observe that RF, DT and KNN are performing well with the union set if the number of features is more and gaining better score with the quorum set in case the number of features is very less. GNB scores better with the quorum set, except for three features which are vice versa. LR is behaving somehow different; as an example, in the span of 5-14 features, the quorum set beats the union, while vice versa for below five features. From Figures 2.3(f)-2.3(j), we observe that all classifiers are performing well with the quorum set compared to the union set except for LR in the range of 13-19 features.
From the performance comparison of different classifiers with the union and quorum sets, we can claim that the quorum set is better for a smaller number of features in the major cases.
So we compare F-score of all five classifiers using the quorum features against the result of individual feature sets on both the datasets. Table 2.2 is a comparison using UNSW-NB15 and Table 2.3 is using NSL-KDD. The results in bold shows top scores. Table 2.2 show that RF, DT and KNN classifiers are outperforming using the quorum feature set in the range of 1-6, LR performs better in the range of 7-12 features of quorum, and GNB is higher scorer for 10 and 12 features of quorum. Table 2.3 highlights that RF and DT are out-performers in the range of 1-5 features with a quorum set, KNN and GNB are higher scorers at the very minimum number of features with a quorum set, while LR is a better performer in the range of 12-19 features and as well as in the range of 7-10 features with quorum set.

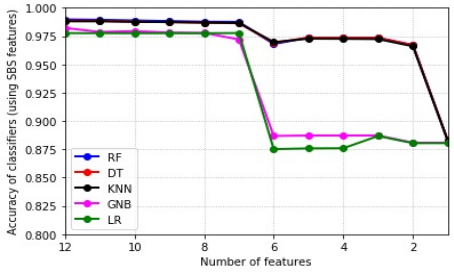
Also, we compared RF, DT and KNN classifiers' AUC scores using the quorum feature set against the two closely related works on the NSL-KDD dataset. The comparison result is shown in Table 2.4; our method outperforms state-of-the-art. To see the execution time taken by each



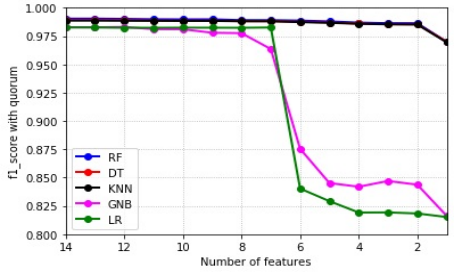
(a) Cross validation score of subset features.



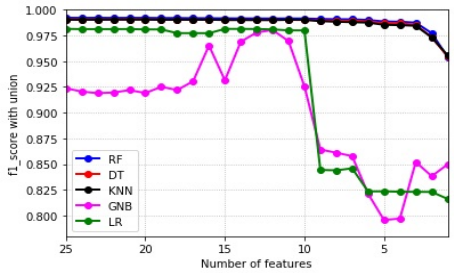
(b) Accuracy score of each subset using RF on test set



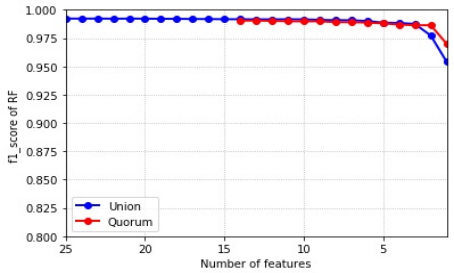
(c) Accuracy score of different classifiers with SBS subset.



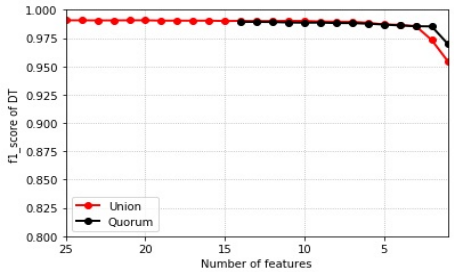
(d) F-score of classifiers with quorum features.



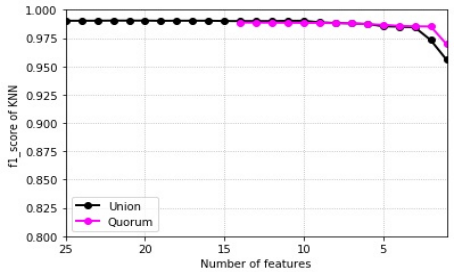
(e) F-score of classifiers with union features.



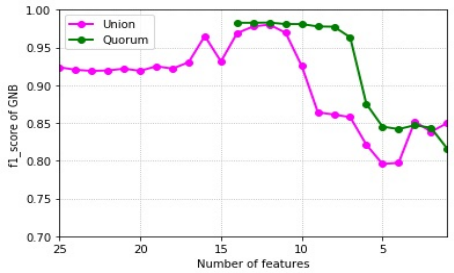
(f) F-score of RF with quorum/union features.



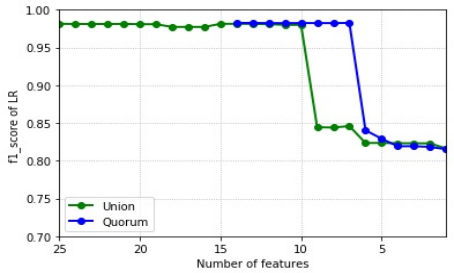
(g) F-score of DT with quorum/union features.



(h) F-score of KNN with quorum/union features.

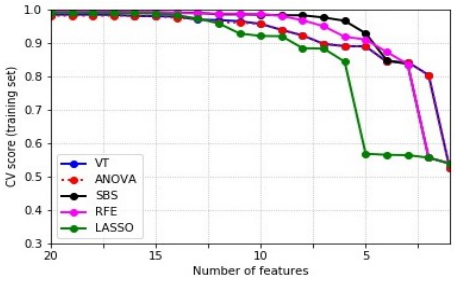


(i) F-score of GNB with quorum/union features.

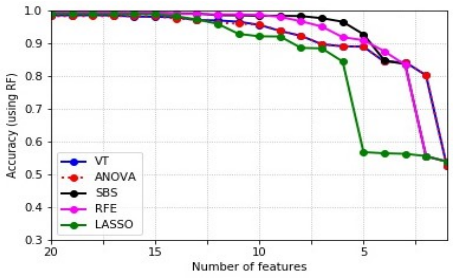


(j) F-score of LR with quorum/union features.

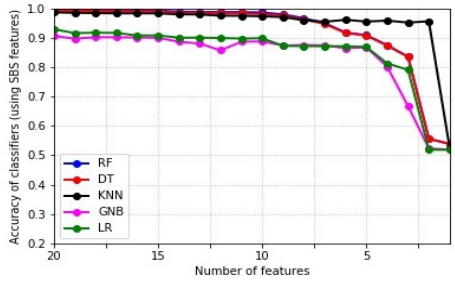
Figure 2.2: Results using UNSW-NB15 dataset.



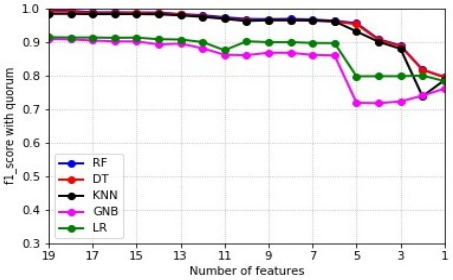
(a) Cross validation score of subset features.



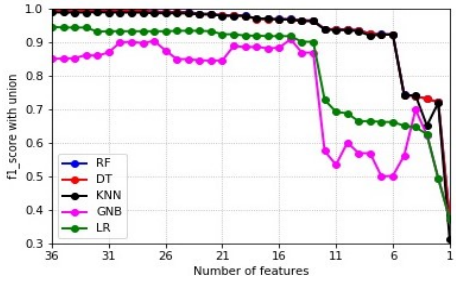
(b) Accuracy score of each subset using RF on test set



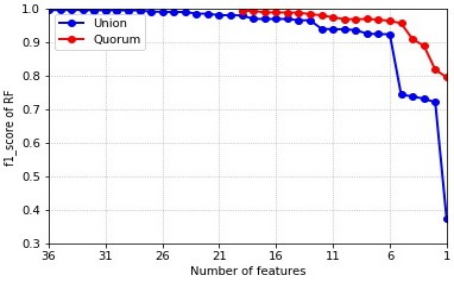
(c) Accuracy score of different classifiers with SBS subset.



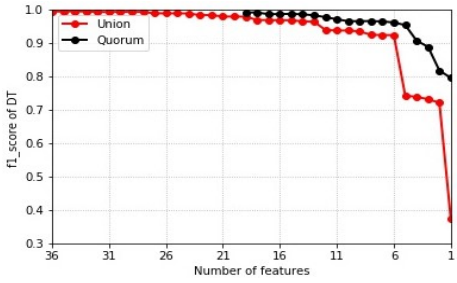
(d) F-score of classifiers with quorum features.



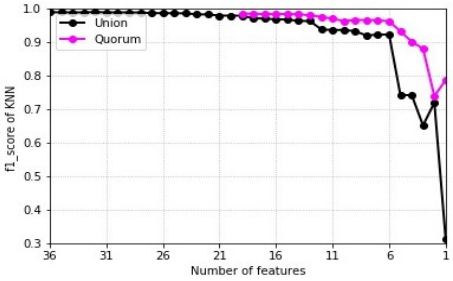
(e) F-score of classifiers with union features.



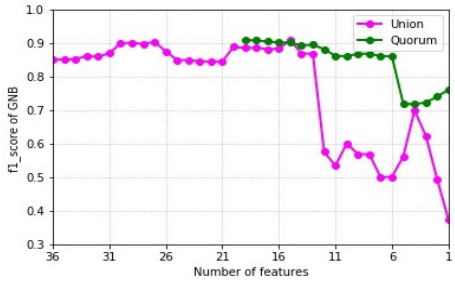
(f) F-score of RF with quorum/union features.



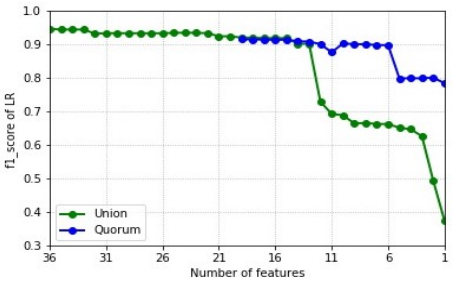
(g) F-score of DT with quorum/union features.



(h) F-score of KNN with quorum/union features.



(i) F-score of GNB with quorum/union features.

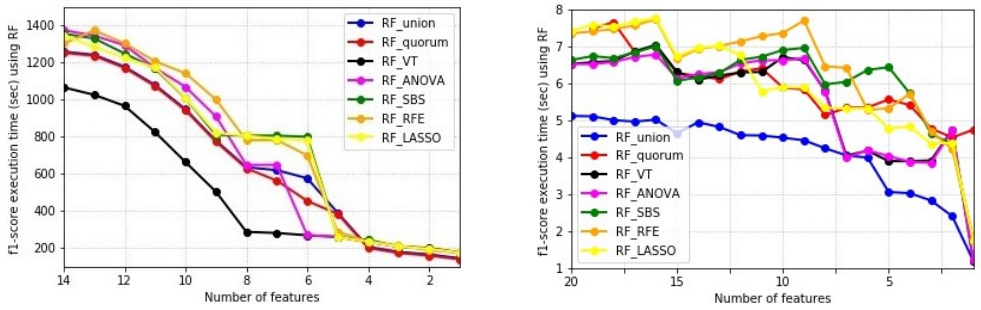


(j) F-score of LR with quorum/union features.

Figure 2.3: Results using NSL-KDD dataset.

Table 2.2: F-score comparison of individual methods feature set vs quorum set using UNSW-NB15.

Classifiers vs No. of feature	12	11	10	9	8	7	6	5	4	3	2	1
RF_VT	99.02%	98.99%	98.98%	98.98%	98.91%	98.81%	98.64%	98.58%	98.41%	98.37%	95.37%	95.37%
RF_ANOVA	98.92%	98.87%	98.86%	98.87%	98.83%	98.82%	98.67%	98.60%	98.51%	98.45%	98.37%	96.91%
RF_SBS	98.96%	98.95%	98.88%	98.82%	98.77%	98.75%	96.73%	97.25%	97.24%	97.23%	96.52%	83.61%
RF_RFE	98.31%	98.11%	98.10%	97.80%	97.76%	96.36%	87.44%	84.42%	84.09%	84.60%	84.28%	81.53%
RF_LASSO	98.24%	98.23%	98.24%	98.24%	98.23%	98.26%	83.93%	82.82%	81.82%	81.83%	81.73%	81.42%
RF_Quorum	**99.02%**	**98.99%**	**98.98%**	**98.98%**	**98.91%**	**98.91%**	**98.87%**	**98.80%**	**98.69%**	**98.64%**	**98.64%**	**97.00%**
DT_VT	**99.02%**	**98.99%**	**98.98%**	**98.98%**	**98.90%**	98.80%	98.58%	98.50%	98.38%	98.35%	95.37%	95.36%
DT_ANOVA	98.92%	98.87%	98.86%	98.87%	98.83%	98.82%	98.67%	98.60%	98.51%	98.45%	98.36%	96.91%
DT_SBS	98.87%	98.85%	98.79%	98.73%	98.68%	98.64%	96.83%	97.25%	97.24%	97.23%	96.52%	83.61%
DT_RFE	98.31%	98.11%	98.10%	97.80%	97.76%	96.36%	87.44%	84.42%	84.09%	84.60%	84.28%	81.53%
DT_LASSO	98.24%	98.23%	98.24%	98.24%	98.23%	98.26%	83.93%	82.82%	81.82%	81.83%	81.73%	81.42%
DT_Quorum	98.92%	98.87%	98.86%	98.87%	98.83%	**98.82%**	**98.77%**	**98.70%**	**98.61%**	**98.55%**	**98.53%**	**97.01%**
KNN_VT	**99.02%**	**98.99%**	**98.98%**	**98.98%**	**98.90%**	98.80%	98.61%	98.50%	98.45%	98.35%	95.27%	95.26%
KNN_ANOVA	98.92%	98.87%	98.86%	98.87%	98.83%	**98.82%**	98.67%	98.60%	98.51%	98.45%	98.35%	96.91%
KNN_SBS	98.86%	98.82%	98.80%	98.80%	98.75%	98.74%	98.69%	98.65%	96.85%	97.12%	96.41%	83.60%
KNN_RFE	98.31%	98.11%	98.10%	97.80%	97.76%	96.36%	87.44%	84.42%	84.09%	84.60%	84.28%	81.53%
KNN_LASSO	98.24%	98.23%	98.24%	98.24%	98.23%	98.26%	83.93%	82.82%	81.82%	81.83%	81.73%	81.42%
KNN_Quorum	98.86%	98.85%	98.84%	98.84%	98.81%	98.81%	**98.73%**	**98.67%**	**98.58%**	**98.54%**	**98.53%**	**96.95%**
GNB_VT	92.43%	93.37%	95.03%	97.70%	**98.41%**	**98.47%**	**98.37%**	97.68%	**98.36%**	**97.72%**	81.39%	81.39%
GNB_ANOVA	91.63%	92.04%	92.58%	94.61%	98.11%	98.28%	98.28%	**98.28%**	97.40%	96.74%	**95.92%**	**88.73%**
GNB_SBS	98.01%	**98.33%**	98.03%	97.68%	97.74%	97.56%	97.51%	96.97%	84.78%	84.79%	83.48%	83.48%
GNB_RFE	98.25%	97.90%	97.97%	97.80%	97.76%	96.36%	84.70%	84.42%	84.09%	84.60%	83.38%	81.53%
GNB_LASSO	98.24%	97.90%	97.97%	**97.85%**	97.81%	97.16%	83.93%	82.82%	81.82%	81.83%	81.73%	81.42%
GNB_Quorum	**98.31%**	98.11%	**98.10%**	97.80%	97.76%	96.36%	87.54%	84.52%	84.19%	84.70%	84.38%	81.63%
LR_VT	98.03%	98.03%	98.03%	97.97%	97.97%	97.95%	**97.85%**	97.67%	**97.66%**	**97.67%**	81.39%	81.39%
LR_ANOVA	97.73%	97.70%	97.69%	97.69%	97.69%	97.67%	97.58%	97.57%	97.61%	97.64%	**97.67%**	**86.39%**
LR_SBS	97.93%	97.74%	97.74%	97.74%	97.74%	97.74%	97.74%	**97.77%**	84.77%	84.71%	83.38%	83.38%
LR_RFE	97.76%	97.76%	97.75%	97.75%	97.75%	96.36%	85.61%	84.42%	84.09%	84.60%	83.28%	81.53%
LR_LASSO	97.76%	97.76%	97.75%	97.75%	97.75%	97.77%	83.93%	82.82%	81.82%	81.83%	81.73%	81.42%
LR_Quorum	**98.24%**	**98.23%**	**98.24%**	**98.24%**	**98.23%**	**98.26%**	84.03%	82.92%	81.92%	81.93%	81.83%	81.52%



(a) Execution time using RF on UNSW-NB15 dataset.

(b) Execution time using RF on NSL-KDD dataset.

Figure 2.4: Execution time comparison of different feature sets with RF.

feature set, we logged F-score execution time of RF classifier with different feature sets; the comparison result is provided in Figure 2.4; we observe that RF with VT consumes less time for higher iterations on UNSW-NB15, while it consumes an average time using NSl-KDD.

2.4 Conclusion

Feature selection is necessary for reducing the computational complexity of training and testing an algorithm, but much intensive analysis is needed to identify an optimal subset in advance. Different selectors identify different feature sets, which certain feature sets may perform better with certain classifiers and vice versa for anomaly detection. As we observed in our experiments,

Table 2.3: F-score comparison of individual methods feature set vs quorum set using NSL-KDD.

Classifiers vs No. of feature	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
RF_VT	98.43%	98.42%	98.38%	98.04%	97.92%	97.76%	96.89%	96.84%	96.48%	95.56%	93.64%	92.13%	89.52%	88.81%	88.70%	84.11%	83.98%	79.85%	36.96%
RF_ANOVA	98.15%	98.15%	98.16%	98.14%	98.17%	97.34%	97.08%	96.08%	95.84%	95.56%	93.68%	92.15%	89.48%	88.84%	88.69%	84.09%	83.95%	79.83%	36.96%
RF_SBS	99.31%	**99.34%**	**99.34%**	99.12%	99.13%	**99.00%**	**98.97%**	98.40%	98.38%	98.27%	**98.20%**	**98.13%**	**97.50%**	**96.47%**	92.56%	84.55%	83.31%	43.31%	40.62%
RF_RFE	**99.35%**	99.29%	99.20%	**99.24%**	**99.20%**	98.84%	98.73%	**98.72%**	**98.66%**	**98.52%**	97.87%	96.56%	95.01%	91.64%	90.75%	87.39%	83.47%	43.35%	40.61%
RF_LASSO	98.92%	98.83%	98.82%	98.80%	98.72%	98.16%	97.12%	95.88%	92.74%	91.97%	91.89%	88.41%	88.32%	84.13%	45.62%	44.97%	44.74%	43.35%	40.61%
RF_Quorum	99.23%	99.21%	98.76%	98.77%	98.73%	98.69%	98.24%	97.90%	97.29%	96.78%	96.73%	96.80%	96.62%	96.25%	**95.52%**	**90.78%**	**88.87%**	**81.72%**	**79.48%**
DT_VT	98.41%	98.45%	98.39%	98.00%	97.91%	97.79%	96.92%	96.79%	96.45%	95.47%	93.64%	92.11%	89.48%	88.84%	88.68%	84.08%	84.00%	79.87%	36.96%
DT_ANOVA	97.97%	98.01%	98.01%	98.00%	98.02%	97.05%	96.85%	95.81%	95.59%	95.30%	93.52%	92.05%	89.42%	88.78%	88.62%	84.03%	83.92%	79.87%	36.96%
DT_SBS	99.06%	**99.09%**	**99.09%**	98.88%	98.89%	**98.76%**	**98.72%**	98.09%	98.10%	97.99%	**97.91%**	**97.78%**	**97.16%**	**96.19%**	92.39%	84.48%	83.35%	43.89%	40.60%
DT_RFE	**99.15%**	99.02%	99.00%	**98.97%**	**98.95%**	98.48%	98.40%	**98.32%**	**98.24%**	**98.06%**	97.54%	96.20%	94.67%	91.53%	90.65%	87.22%	83.35%	43.89%	40.60%
DT_LASSO	98.61%	98.53%	98.44%	98.45%	98.38%	97.87%	96.80%	95.52%	92.63%	91.88%	91.77%	88.40%	88.30%	84.25%	45.93%	45.25%	45.25%	43.89%	40.60%
DT_Quorum	99.01%	99.01%	98.49%	98.54%	98.59%	98.46%	98.17%	97.60%	96.97%	96.40%	96.41%	96.43%	96.32%	96.03%	**95.23%**	**90.59%**	**88.75%**	**81.66%**	**79.48%**
KNN_VT	98.29%	98.27%	98.25%	97.74%	97.68%	97.45%	96.60%	96.49%	96.17%	95.30%	93.52%	92.05%	89.42%	88.78%	88.62%	84.03%	83.92%	79.87%	36.96%
KNN_ANOVA	97.97%	98.01%	98.01%	98.00%	98.02%	97.05%	96.85%	95.81%	95.59%	95.30%	93.52%	92.05%	89.42%	88.78%	88.62%	84.03%	83.92%	79.87%	36.96%
KNN_SBS	**98.66%**	**98.66%**	**98.62%**	**98.41%**	**98.41%**	**98.32%**	**98.32%**	**97.85%**	**97.85%**	**97.78%**	**97.63%**	**97.55%**	**97.00%**	**97.08%**	**96.33%**	**95.92%**	95.14%	**95.58%**	41.45%
KNN_RFE	98.45%	98.34%	98.28%	98.28%	98.27%	97.93%	97.88%	97.47%	97.44%	97.28%	96.98%	95.92%	95.41%	96.07%	95.48%	95.80%	95.14%	95.58%	41.45%
KNN_LASSO	98.08%	98.03%	97.67%	97.62%	97.55%	97.31%	96.46%	95.80%	96.34%	96.41%	96.13%	95.92%	95.98%	95.28%	95.98%	95.75%	**95.76%**	95.58%	41.45%
KNN_Quorum	98.33%	98.32%	98.23%	98.25%	98.24%	98.17%	97.84%	97.41%	96.83%	96.16%	96.47%	96.42%	96.37%	96.09%	93.08%	90.01%	87.92%	73.78%	**78.60%**
GNB_VT	97.99%	97.98%	98.10%	97.76%	97.66%	97.58%	96.85%	96.69%	96.22%	95.33%	93.66%	92.08%	89.65%	88.88%	78.02%	83.32%	80.28%	76.17%	31.44%
GNB_ANOVA	97.85%	97.86%	97.86%	97.86%	97.86%	96.97%	96.80%	95.98%	95.80%	95.33%	93.66%	92.08%	89.65%	88.88%	78.02%	83.32%	80.28%	76.17%	31.44%
GNB_SBS	**98.66%**	**98.66%**	**98.62%**	**98.41%**	**98.41%**	**98.32%**	**98.32%**	**97.85%**	**97.85%**	**97.78%**	**97.63%**	**97.55%**	**97.00%**	**97.08%**	**96.33%**	**95.92%**	**95.14%**	**95.58%**	41.45%
GNB_RFE	89.66%	90.16%	90.14%	90.04%	89.95%	88.60%	87.98%	85.56%	88.59%	88.70%	87.29%	87.44%	87.31%	86.34%	86.62%	79.49%	63.31%	37.76%	37.33%
GNB_LASSO	88.02%	88.05%	88.76%	89.41%	89.21%	87.26%	87.26%	87.37%	87.19%	87.64%	84.60%	79.06%	79.22%	53.36%	40.10%	39.33%	39.32%	37.76%	37.33%
GNB_Quorum	90.81%	90.70%	90.39%	90.12%	90.12%	89.20%	89.45%	88.08%	86.12%	86.00%	86.74%	86.72%	86.08%	85.95%	71.86%	71.73%	72.25%	73.95%	**76.09%**
LR_VT	90.56%	90.68%	90.63%	89.36%	89.35%	89.30%	88.88%	88.56%	88.58%	86.57%	86.76%	87.00%	87.02%	87.42%	**87.23%**	79.05%	**80.96%**	67.10%	36.96%
LR_ANOVA	90.99%	88.30%	87.76%	88.07%	90.66%	87.40%	87.27%	86.93%	86.58%	86.57%	86.76%	87.00%	87.02%	87.42%	87.23%	79.05%	80.96%	67.10%	36.96%
LR_SBS	89.39%	90.72%	90.40%	89.94%	90.01%	89.97%	89.94%	88.73%	**88.78%**	88.48%	87.57%	88.48%	67.41%	65.74%	66.44%	67.98%	63.31%	37.76%	37.33%
LR_RFE	89.66%	90.16%	90.14%	90.04%	89.95%	88.60%	87.98%	85.56%	88.59%	88.70%	87.29%	87.44%	87.31%	86.34%	86.62%	79.49%	63.31%	37.76%	37.33%
LR_LASSO	90.78%	90.79%	90.78%	90.57%	90.60%	88.13%	87.89%	87.94%	88.10%	87.93%	87.84%	82.07%	82.04%	79.74%	39.71%	38.92%	38.91%	37.32%	37.32%
LR_Quorum	**91.36%**	**91.28%**	**91.27%**	**91.19%**	**91.25%**	**90.77%**	**90.68%**	**89.93%**	87.49%	**90.14%**	**89.89%**	**89.90%**	**89.64%**	**89.62%**	79.73%	**79.76%**	79.76%	**79.95%**	**78.31%**

Table 2.4: AUC comparison of the proposed method and state-of-the-art on NSL-KDD.

Method	**U2R**	**Probe**
Pang et al. [52]	97.0%	99.0%
Pang et al.[53]	NA	96.6%
RF with quorum set	99.8%	99.5%
DT with quorum set	99.7%	99.3%
KNN with quorum set	99.7%	99.4%

relying on a single feature set selected with a single technique may result in a risky decision. To minimize the potential risk, we proposed an ensemble learning approach to combine the results of independent selection methods with the help of union and quorum techniques, with the goal of identifying an optimal feature set. Our study results highlight that union and quorum techniques generated competitive feature sets. The performance of the reduced feature set was approximately the same as the performance of the entire feature set for network anomaly detection. The experimental results on the UNSW-NB15 dataset show that RF with union and quorum feature sets yields 99% and 99.02% F-score with minimum 6 and 12 features, respectively. The results on the NSL-KDD dataset show that RF with union and quorum yields 99.34% and 99.21% F-score with minimum 28 and 18 features.

Chapter 3

Network Anomaly Detection Using Ensemble Learning Approach [1] [2]

3.1 Introduction

Network attacks are a frequent occurrence in today's digitally connected society. They are unauthorized access to an organizational network's digital assets. Firewalls provide security to the network but have limited capability to analyze network packets to identify various malicious activities and block them. On the other hand, Network Intrusion Detection Systems (NIDS) protect network infrastructure by monitoring and analyzing network traffic data. There are mainly two types of NIDS based on their detection techniques. One is misuse detection (signature-based) technique, and the other is anomaly-based detection technique [54]. The misuse detection technique has a signature database for known attacks, and it compares the new traffic with those already known attacks' signatures. If there is any match, the alarm is generated for the attack. On the other hand, anomaly-based detection techniques distinguish attacks by identifying deviations in traffic from normal patterns. The misuse technique is fast in the detection of known attacks and produces a lower false alarm rate. However, it requires a frequent manual update of the signature database. Anomaly-based techniques are good in identifying unknown attacks, but it retains a higher false alarm rate.
Machine Learning algorithms are extensively used in anomaly-based IDS. Different ML algorithms such as support vector SVM [55, 56], DT [57, 58], and RF [59, 60] have been employed to classify normal and abnormal traffic. In the purview of IDS, many studies used a combination of classifiers, and it is assumed that an ensemble learner performs better than an individual classifier due to several justifications such as computational, statistical and representational [61]. In the ensemble learning approach, several ML algorithms' outputs are combined together to obtain a better result. If weak classifiers with overall low performance but good performance for some particular class are put together, then theoretically, it is possible to result in a good performance by combining the results of more than one weak classifier. There are various ensemble methods, such as Stacking, Bagging, and Boosting. Stacking, combines different classification by a meta-classifier [62], the base classifiers are trained with the whole set of training data, and then the meta-model is trained on the outputs of the base classifiers. Bagging refers to the approach in which the same classifier is trained on different subsets of the same data. Boosting

[1] Doreswamy and Mohammad Kazim Hooshmand. **Using Ensemble Learning Approach to Identify Rare Cyber-Attacks in Network Traffic Data**. *IEEE International Conference on Advanced Computer Science and Information Systems (ICACSIS) 2020*. 17 October 2020. pp.141-146.

[2] Doreswamy and Mohammad Kazim Hooshmand. **Network Traffic Data Classification Using Machine Learning Algorithms**. *International conference on computational intelligence and applications (ICCIA-2019)*, 09 August 2019, pp. 123-128.

refers to a family of algorithms with the ability to transform weak learners into strong learners. This study exploits the advantages of imbalance processing and ensemble learners in Network Anomaly Detection Systems (NADS) through intensive comparative studies among various base learners and different combination methods. Our main contributions are as follows:

- We present a new approach to imbalance processing using SMOTE oversampling technique and a cluster-based undersampling with the K-means algorithm (SKM). We use SMOTE to oversample the minority class and K-means algorithm to perform cluster-based undersampling on the majority class. The outcome of this approach is to balance data with minimum loss of information and without increasing data size.
- We propose an ensemble-based network anomaly detection model, SKM-XGB, which combines the class imbalance method SKM and XGBoost algorithm to identify malicious traffic in imbalanced network traffic data with a high detection rate.

3.2 The Proposed Method

The main steps of our methods are data preprocessing, imbalance processing, algorithms parameter optimization, training the model based on the best parameters, and evaluating the model. In this work, we use full UNSW-NB15 and NSL-KDD. Table 3.1 gives the number of instances for each class split into train-set, test-set, and validation-set on both datasets.

3.2.i. Data Preprocessing

We dropped redundant meaningless features [63] such as switch information (srcip, sport, dstip, dsport) and time- related ones (stime, ltime) from the UNSW-NB15 prior to the data preprocessing step. The data preprocessing consists of three main steps: one-hot encoding, data standardization, and feature selection. UNSW-NB15 and NSL-KDD have three nominal features each. Nominal features of UNSW-NB15 are "proto", "state" and "service", and nominal features of NSL-KDD are "protocol type", "service", and "flag". After applying one-hot encoding, the UNSW-NB15 feature dimensions increase to 202, and the feature dimensions of NSL-KDD increase to 121. Similarly, one-hot encoding is applied on the class label of both datasets.

Table 3.1: The number of instances in each class of UNSW-NB15 and NSL-KDD datasets.

Dataset	Class	Train-set (70%)	Test-set (20%)	Validation-set (10%)	Total
UNSW-NB15	Normal	1,559,255	436,755	222,751	2,218,761
	Generic	151,430	42,418	21,633	215,481
	Exploits	31,291	8,764	4,470	44,525
	Fuzzers	17,039	4,773	2,434	24,246
	DoS	11,491	3,220	1,642	16,353
	Reconnaissance	9,830	2,753	1,404	13,987
	Analysis	1,881	527	269	2,677
	Backdoor	1,637	458	234	2,329
	Shellcode	1,061	298	152	1,511
	Worms	123	34	17	174
	Total	**1,785,038**	**500,000**	**255,006**	**2,540,044**
NSL-KDD	Normal	54,150	15,168	7,736	77,054
	DoS	37,517	10,508	5,360	53,385
	Probe	9,893	2,772	1,412	14,077
	R2L	2,634	738	377	3,749
	U2R	177	50	25	252
	Total	**104,371**	**29,236**	**14,910**	**148,517**

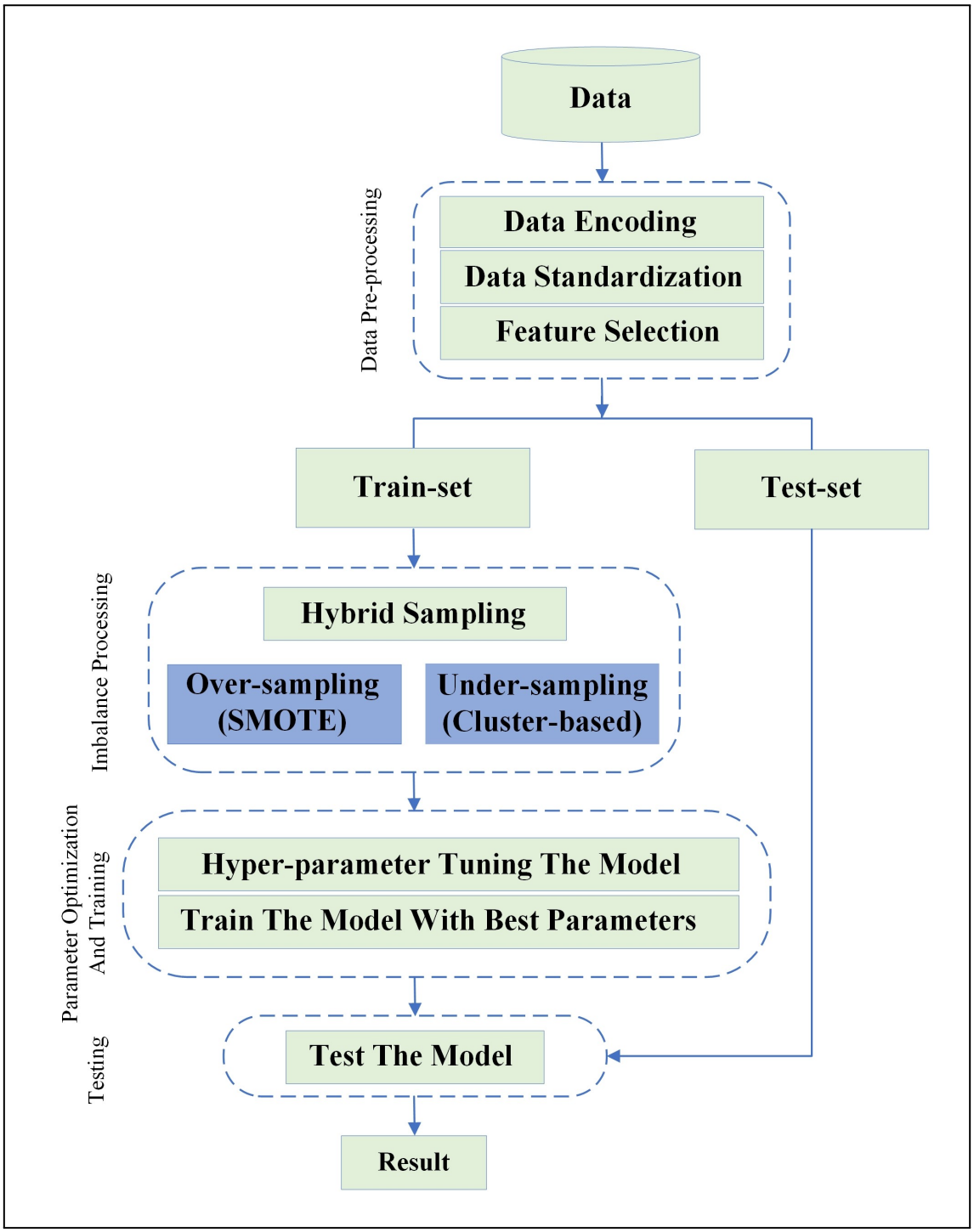


Figure 3.1: The proposed method.

Next, we standardize all the remaining features according to Equation 3.1 and normalize them to Normal Distribution, also called the Gaussian Distribution, with a mean of 0 and a variance of 1.

$$x^{'} = \frac{x - \mu}{\sigma} \tag{3.1}$$

where x' is the normalized features, x is the original feature, μ is the mean, and σ is the standard deviation.

Finally, we applied feature selection using Denoising Autoencoder (DAE) on both datasets and selected 15 features for each one. Table 3.2 shows 15 selected features' names of both datasets.

Table 3.2: Selected features of UNSW-NB15 and NSL-KDD datasets.

Dataset	**Selected Features**
UNSW-NB15	dtcpb, stcpb, service_-, dmeansz, dload, sload, service_dns, smeansz, trans_depth, sttl, djit,service_ftp-data, ct_ftp, ct_state_ttl, sloss
NSL-KDD	service_http, dst_host_srv_rerror_rate, duration, dst_host_same_srv_rate, protocol_type_udp, same_srv_rate, dst_host_serror_rate, srv_rerror_rate, logged_in, service_telnet, dst_host_srv_serror_rate, service_other, flag_SF, num_root, srv_count

3.2.ii. Data Imbalance Processing

Reference to Table 3.1, number of instances are too small for some class. As example *Worms*, *Shellcode*, and *Backdoor* have few samples in UNSW-NB15 and likewise, the number of samples for *U2R* and *R2L* are less in NSL-KDD compared to the other classes. This class imbalance affects the model performance. We present a combined oversampling technique using SMOTE and a cluster-based undersampling technique using the K-means algorithm to balance the data. Oversampling alone increases the size of data and ultimately increases the computational cost of the model. On the other hand, undersampling decreases the size of data but causes information loss by eliminating some informative samples. So, we used a combination of both techniques. For the minority class, we use SMOTE oversampling technique. SMOTE works by picking random points between minority class instances and its k-nearest neighbours as new samples. In the clustering-based undersampling technique, we divide the data belonging to each majority class into 10 clusters and randomly choose some portion from each cluster so that the sum of all taken portions has to be equal to the selected threshold that the minority class has been oversampled to. Algorithm 1 describes our new imbalance processing method SKM.

3.3 Experimental Results and Analysis

We implemented the proposed NAD model in Python 3.6.13 and conducted the experiments on a workstation with Windows 11 Pro 64 bits operating system.

3.3.i. Hyperparameter Optimization

We performed a parameter optimization for XGBoost and the other models used for comparison purposes in this work on both datasets. However, it requires much time and resources to test all combination possibilities of parameters with a larger range of values. We optimized three parameters of XGBoost using random search cross-validation and kept the remaining parameters

Table 3.3: The parameters and best parameters of models.

Model	Parameters	Best Parameters with UNSW-NB15	Best Parameters with NSL-KDD
RF	n_estimators= np.linspace(10, 200).astype(int), max_depth= [None] + list(np.linspace(3, 30).astype(int)), max_features= ['auto', 'sqrt', None], max_leaf_nodes= [None] + list(np.linspace(10, 50, 500).astype(int)), min_samples_split= [2, 5, 10], bootstrap= [True, False]	n_estimators= 25, max_depth= 20, max_features= 'sqrt', max_leaf_nodes= 38, min_samples_split= 10, bootstrap= False	n_estimators= 157, max_depth= 27, max_features= None, max_leaf_nodes= 48, min_samples_split= 5, bootstrap= True
MLP	hidden_layer_sizes= [(10,),(20,)], activation= ['tanh', 'relu'], solver= ['sgd', 'adam'], alpha= [0.0001, 0.05], learning_rate= ['constant','adaptive']	hidden_layer_sizes= (10,), activation= 'tanh', solver=' adam', alpha= 0.05, learning_rate= 'constant'	hidden_layer_sizes= (20,), activation= 'relu', solver=' adam', alpha= 0.05, learning_rate= 'adaptive'
KNN	n_neighbors= [3,5,11,19], weights= ['uniform','distance'], p= [1,2], algorithm= ['ball_tree', 'kd_tree', 'brute']	n_neighbors= 19, weights= 'uniform', p= 1, algorithm= 'brute'	n_neighbors= 19, weights= 'uniform', p= 1, algorithm= 'brute'
AdaBoost	n_estimators= [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 20], learning_rate= [(0.97 + x / 100) for x in range(0, 8)], algorithm= ['SAMME', 'SAMME.R']	n_estimators= 3, learning_rate= 1.03, algorithm= 'SAMME.R'	n_estimators= 9, learning_rate= 1.03, algorithm= 'SAMME.R'
LGBM	n_estimators= [50, 100, 200, 300, 500], learning_rate= [0.01, 0.05, 0.1, 0.15], max_depth= np.arange(1,7,2)	n_estimators= 300, learning_rate= 0.01, max_depth= 5	n_estimators= 300, learning_rate= 0.01, max_depth= 5
XGB	n_estimators= [50, 100, 200, 300, 500], learning_rate= [0.01, 0.05, 0.1, 0.15], max_depth= np.arange(1,7,2)	n_estimators= 500, learning_rate= 0.05, max_depth= 5	n_estimators= 500, learning_rate= 0.1, max_depth= 5

Algorithm 1 SKM

Required:

C_n = The total number of classes;
Training set $S = S_i, i = 1, 2, 3, ...C_n$;
$|S| = N$; #All samples
$cl = 10$ #numbr of clusters

Output:

a balanced training set $S^{'}$;

$Avg = int(N/C_n)$
for $i \leftarrow 1$ to C_n **do**
  if $|Si| < Avg$ **then**
    $S_i^{'} = SMOTE(S_i, Avg)$ # Apply SMOTE on S_i, so that $|S_i^{'}| = Avg$
  end if
  if $|S_i| > Avg$ **then**
    $K_p = K - means(S_i, cl)$ #Making cl clusters from S_i, $p = 1, 2, 3...., cl$
    for $p \leftarrow 1$ to cl **do**
      if $K_p > Avg/cl$ **then**
        $K_p^{'} = Resample(K_p, Avg/cl)$ #Avg/cl random samples from K_p
      else
        $K_p^{'} = Avg/cl$ #assign whole cluster
    end for
    $S_i^{'} = Merge(K_p^{'})$
  end if
  $S^{'} = Merge(S_i^{'})$
end for
return $S^{'}$

with their default values. The optimized parameters of XGBoost are *n_ estimator*, *learning_ rate* and *max_ depth*. To compare our model's performance with the other ensemble methods, we use three other algorithms, such as RF, Multi-Layer Perceptron (MLP), and KNN, as base classifiers with the Max-voting, Stacking, and Bagging combination methods. Moreover, to demonstrate the effectiveness of our model, we compared it with AdaBoost and LGBM ensemble models as well. The same parameter optimization method with random search cross-validation is used to optimize the other models' parameters. Table 3.3 presents all the models, which are tested, the set of given parameters, and the selected best parameters with respect to both datasets.

3.3.ii. Classification

We conducted binary classification as well as multiclass classification. Table 3.4 presents the results of binary classification for the SKM-XGB model on both datasets; Table 3.5 contains the result of the multiclass classification of SKM-XGB model as well as the results of other ensemble models which we tested for comparative analysis in this study. We compared our

Table 3.4: SKM-XGB binary classification performance on UNSW-NB15 and NSL-KDD datasets.

Model	Dataset	Accuracy %	Precision %	Recall %	F-score %	FAR %	Train_time (sec)	Test_time (sec)
SKM-XGB	UNSW-NB15	99.01	99.07	99.01	99.02	0.58	69	0.30
	NSL-KDD	99.37	99.37	99.37	99.37	0.64	16	0.04

model's multiclass classification results with the results obtained by RF, MLP, KNN, AdaBoost,

LGBM, Max-voting, Stacking, and Bagging. The base classifiers for Max-voting, Stacking, and Bagging have been RF, MLP, and KNN. From binary classification, we observe that SKM-XGB

Table 3.5: SKM-XGB multiclass classification performance on UNSW-NB15 and NSL-KDD datasets.

Dataset	Model	Acc. %	Precision %	Recall %	F-score %	FAR %	Train_time (s)	Test_time (s)
UNSW-NB15	RF	98.82	98.10	96.83	97.25	0.52	3.48	0.48
	MLP	98.85	97.95	96.37	96.96	0.23	48.22	**0.17**
	KNN	98.94	97.82	96.79	97.23	0.51	0.08	1882.16
	Voting_RF_MLP_KNN	98.84	98.07	96.68	97.17	0.52	55.24	966.18
	Stacking_RF_MLP_KNN	98.69	95.46	95.84	95.42	1.27	1265.88	825.59
	Bagging_RF	97.79	97.06	95.40	94.71	10.04	6.95	2.96
	Bagging_MLP	97.64	93.44	94.25	93.15	4.63	0.39	1.31
	Bagging_KNN	88.18	88.95	87.35	81.45	87.35	**0.06**	6.30
	AdaBoost	98.36	98.45	95.56	96.02	**0.17**	2.87	0.37
	LGBM	98.91	98.30	97.01	97.47	0.34	31.63	8.15
	SKM-XGB	**99.08**	**98.46**	**97.49**	**97.84**	0.61	721.72	3.41
NSL-KDD	RF	96.62	96.19	93.83	94.60	0.85	10.86	0.11
	MLP	96.87	96.31	94.33	94.98	0.94	63.48	**0.01**
	KNN	98.75	98.12	97.85	97.93	0.61	0.04	40.59
	Voting_RF_MLP_KNN	98.08	97.42	96.58	96.85	0.71	122.08	39.77
	Stacking_RF_MLP_KNN	98.47	97.67	97.41	97.49	0.71	527.36	37.65
	Bagging_RF	86.74	89.11	76.75	81.14	5.35	13.42	0.58
	Bagging_MLP	89.22	87.34	80.67	83.63	6.40	0.36	0.07
	Bagging_KNN	86.85	87.23	72.38	73.82	11.87	**0.03**	0.41
	AdaBoost	82.84	85.04	69.89	75.24	7.51	4.10	0.06
	LGBM	97.99	97.66	96.37	96.78	0.41	10.19	0.30
	SKM-XGB	**99.57**	**99.26**	**99.22**	**99.24**	**0.20**	190.61	0.16

scores 99.01% and 99.37% detection rate for UNSW-NB15 and NSL-KDD, respectively, while the FAR are 0.58% and 0.64% respectively. In multiclass classification, SKM-XGB outperforms all the other ensemble models in all metrics such as Accuracy, Precision, Recall, F-score, and FAR, while in the training time, Bagging with KNN is the best, and for the testing time, MLP is the best among all models on both datasets.

To demonstrate the effectiveness of the SKM-XGB model, we compared it with some of the state-of-the-art intrusion detection methods, as given in table 3.6.

Table 3.6: A comparison of the proposed model and previous works.

Dataset	Classification	Model	Acc. %	DR %	F-score %	FAR %
UNSW-NB15	Multiclass	RCNF [64]	95.98	-	-	4.02
		SCM3+RF [60]	95.87	97.40	-	6.90
		LightGBM+ADASYN [65]	89.56	-	-	-
		In this work	**99.08**	**97.49**	**97.84**	**0.61**
	Binary	RepTree [66]	88.95	-	-	-
		IG+ANN [57]	97.04	-	-	1.48
		DAE+MLP [63]	98.80	94.43	95.20	**0.57**
		In this work	**99.01**	**99.01**	**99.02**	0.58
NSL-KDD	Multiclass	LightGBM+ADASYN [65]	92.57	-	-	-
		In this work	**99.57**	**99.22**	**99.24**	**0.20**
	Binary	RepTree [66]	89.85	-	-	-
		In this work	**99.37**	**99.37**	**99.37**	**0.64**

3.4 Conclusion

To overcome the data imbalance issue without much loss of information, we proposed a combined method of oversampling using SMOTE and an undersampling technique based on the K-means clustering algorithm. The imbalance handling technique is then integrated with an

ensemble-based model using XGBoost for network anomaly detection. The proposed SKM-XGB performance is evaluated on two commonly used datasets, UNSW-NB15 and NSL-KDD. The experimental results show that SKM-XGB produces a detection rate of 99.01% and 97.49% for binary classification and multiclass classification on UNSW-NB15, respectively. For binary classification and multiclass classification on the NSL-KDD dataset, the model achieves a detection rate of 99.37% and 99.22%, respectively. Moreover, the false alarm rate generated by SKM-XGB is significantly reduced, which is very important for network anomaly detection systems. Comparing SKM-XGB with some previous works shows that our model is superior to the state-of-the-art, which will be promising for future NADS development.

Chapter 4

Deep Learning-based Network Anomaly Detection [1] [2]

4.1 Introduction

The pervasive use of digital devices in today's digitally connected world, especially the IoT paradigm in different aspects of lives [67], brings much information into cyberspace. On the other hand, this overall development and changes cause hidden dangers [68, 69] to digital assets. Hence, protecting digital assets is a critical challenge and motivates network security scholars to research the domain. In recent years, many network security incidents have happened on personal as well as commercial systems with different attack methods [70]; it shows that not only the number of attacks increased but also the forms of attacks increased dramatically as well. The traditional in place security tools as a first-line network security defence, such as encryption methods and firewalls, cannot cope with all types of network security dimensions. Therefore, developing an effective tool to detect network attacks is highly needed. Network Anomaly Detection Systems are promising method to identify new intrusive activities and unauthorized access to digital assets. Moreover, network traffic data is huge, which results in computational costs for anomaly detection [71]. In the real world, network traffic data is an imbalance that causes a delay in the model's convergence and results in biased prediction [72, 73]. Resampling techniques are widely employed methods to balance data. Resampling includes oversampling and undersampling. Each one has its pros and cons. Oversampling is good for keeping all information but increases the size of data, while undersampling decreases the size of data but causes information loss to some extent based on the type and proportion of sampling. The simplest method is Random Oversampling (ROS) and Random Undersampling (RUS). Some of the other representations are BalanceCascade [41], SMOTE [40] and ADASYN [39]. There are many recommendations for oversampling in previous works; for example, in work [42], the authors have proved that oversampling does not cause overfitting, and it is the best way to handle imbalance problems in DL. We propose a feature selection technique using DAE to reduce the data dimensionality to solve the problems mentioned earlier. To handle imbalance issues in network traffic data, we use a combined method of oversampling, ADASYN and a cluster-based undersampling, using the K-means algorithm. The aim of using the combined method is to avoid the issue of data size increase with the help of undersampling and exploits

[1] Mohammad Kazim Hooshmand and Doreswamy. **Machine Learning Based Network Anomaly Detection**. *International journal of recent technology and engineering.* 2019, Vol. 8, Issue 4, pp. 542–548, ISSN: 2277-3878.

[2] Mohammad Kazim Hooshmand and Manjaiaha D. H. **Network Intrusion Detection with 1D Convolutional Neural Networks**. *Digital Technologies Research and Applications.* 2022, Vol. 01, Issue 02, pp. 25–34, ISSN: 2754-5687.

the advantages of keeping as many as informative samples with the help of oversampling. Then, this imbalance processing approach is fused to 1D CNN to perform binary classification and multiclass classification tasks. The performance of the model is evaluated on NSL-KDD and UNSW-NB15 datasets. Our main contributions in this study are:

- We present a combined ADASYN and K-means-based clustering method to handle imbalance issues in the network traffic data.
- We propose a flow-based NADS that employs an integrated imbalance processing method and a 1D CNN model. Our model performs superior to the state-of-the-art in producing detection rates and dramatically reducing false alarm rates. It will be a prominent point for future NADS design and development.

4.2 The Proposed Method

The proposed method mainly consists of data preprocessing, class imbalance handling, classification, and evaluation. Figure 4.1 illustrates the architecture of the model. Table 4.1 gives the number of instances for each class split into train-set, test-set, and validation-set on both datasets.

Table 4.1: The number of instances in each class of UNSW-NB15 and NSL-KDD datasets.

Dataset	Class	Train-set (70%)	Test-set (20%)	Validation-set (10%)	Total
UNSW-NB15	Normal	1,559,255	436,755	222,751	2,218,761
	Generic	151,430	42,418	21,633	215,481
	Exploits	31,291	8,764	4,470	44,525
	Fuzzers	17,039	4,773	2,434	24,246
	DoS	11,491	3,220	1,642	16,353
	Reconnaissance	9,830	2,753	1,404	13,987
	Analysis	1,881	527	269	2,677
	Backdoor	1,637	458	234	2,329
	Shellcode	1,061	298	152	1,511
	Worms	123	34	17	174
	Total	**1,785,038**	**500,000**	**255,006**	**2,540,044**
NSL-KDD	Normal	54,150	15,168	7,736	77,054
	DoS	37,517	10,508	5,360	53,385
	Probe	9,893	2,772	1,412	14,077
	R2L	2,634	738	377	3,749
	U2R	177	50	25	252
	Total	**104,371**	**29,236**	**14,910**	**148,517**

4.2.i. Data Preprocessing

Data preprocessing in this chapter is the same as data preprocessing in section 3.2.i.. First, we dropped unnecessary features such as "srcip", "sport", "dstip", "dsport", "stime", and "ltime" [63] from the UNSW-NB15 dataset prior to the data preprocessing step. Data preprocessing consists of three main steps: one-hot encoding, data standardization, and feature selection. One-hot encoding is a process of converting nominal features into binary vectors [70]. The goal of performing data standardization is to bring down all the features to a common scale without distorting the differences in the range of the values. Feature selection reduces the number of variables to an optimal set by eliminating redundant or unnecessary variables. UNSW-NB15 and NSL-KDD have three nominal features each. Nominal features of UNSW-NB15 are "proto", "state", and "service", and nominal features of NSL-KDD are "protocol type", "service", and "flag". After applying one-hot encoding, the UNSW-NB15 feature dimension increased to 202,

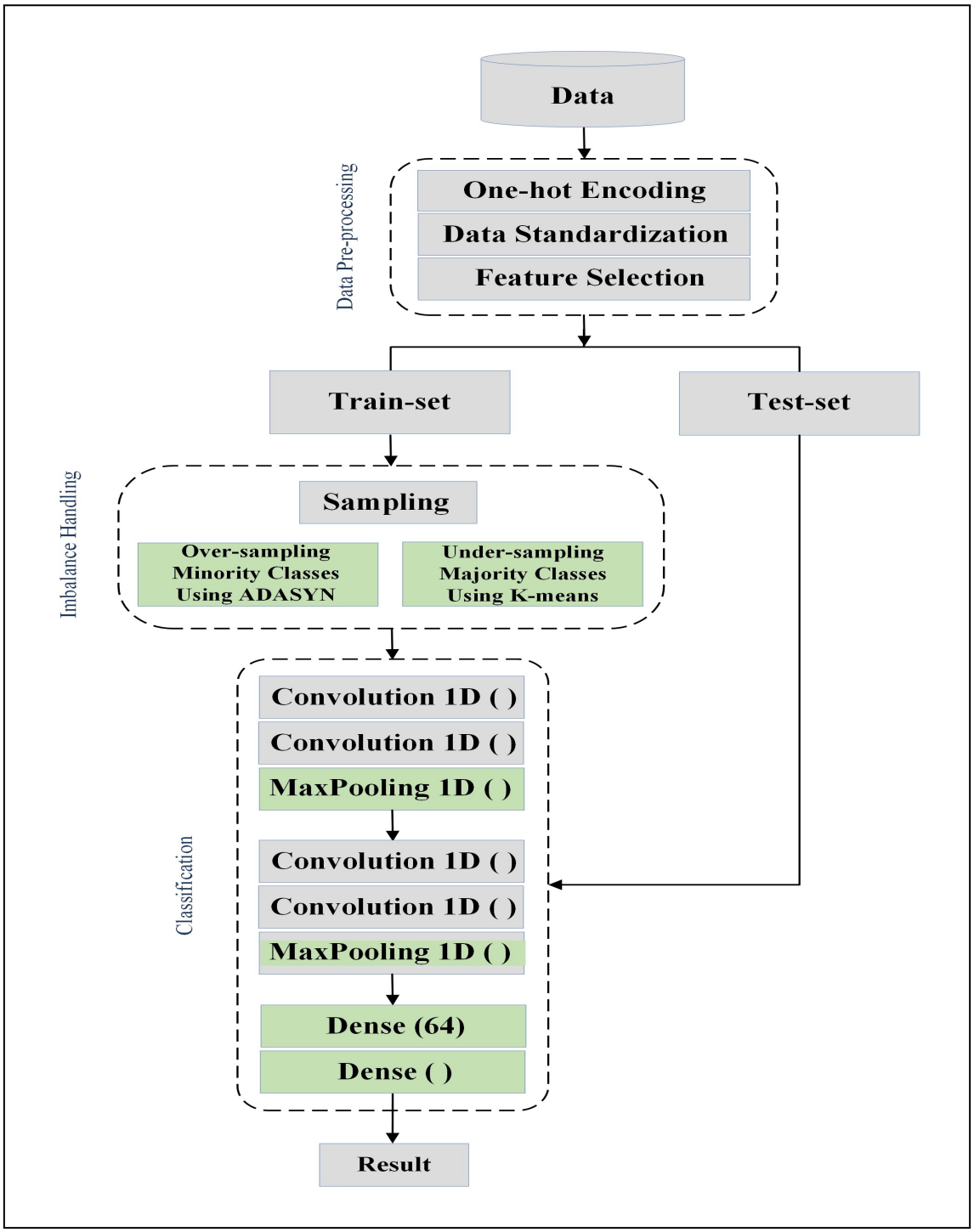


Figure 4.1: The proposed method.

and the feature dimension of NSL-KDD increased to 121. Similarly, one-hot encoding is applied on the class label of both datasets. Next, we standardize all the remaining features according to Equation 4.1 and normalize them to Normal Distribution, also called the Gaussian Distribution, with a mean of 0 and a variance of 1.

$$x^{'} = \frac{x - \mu}{\sigma} \tag{4.1}$$

where x' is the normalized features, x is the original feature, μ is the mean, and σ is the standard deviation.

Finally, we perform feature selection using DAE on both datasets and select the top 15 features for each one. Table 4.2 shows 15 selected features' names of both datasets.

Table 4.2: Selected features of UNSW-NB15 and NSL-KDD datasets.

Dataset	**Selected Features**
UNSW-NB15	dtcpb, stcpb, service_-, dmeansz, dload, sload, service_dns, smeansz, trans_depth, sttl, djit,service_ftp-data, ct_ftp, ct_state_ttl, sloss
NSL-KDD	service_http, dst_host_srv_rerror_rate, duration, dst_host_same_srv_rate, protocol_type_udp, same_srv_rate, dst_host_serror_rate, srv_rerror_rate, logged_in, service_telnet, dst_host_srv_serror_rate, service_other, flag_SF, num_root, srv_count

4.2.ii. Data Imbalance Processing

Reference to Table 4.1, the number of instances is too small for some classes in both datasets. For example, *Worms*, *Shellcode*, and *Backdoor* have few samples in UNSW-NB15, and similarly, the number of samples for *U2R* and *R2L* are smaller in NSL-KDD compared to the other classes. We present a combined oversampling technique using ADASYN and a cluster-based undersampling technique using the K-means algorithm to balance the data. Oversampling alone increases the data size, ultimately increasing the model's computational cost and affecting model accuracy. Undersampling decreases data size but causes information loss by eliminating some informative transactions. To overcome those shortcomings, we use a combination of both resampling techniques. For the minority class, we use the ADASYN technique. ADASYN is an improved version of SMOTE. The key idea behind the use of ADASYN is that it employs a density distribution as a criterion to automatically decide the number of synthetic samples that are required to be generated for each minority example [39]. The way it works is similar to SMOTE, with a minor improvement. It adds some small random values to the points, making them more realistic. So, instead of all the samples being linearly correlated to the parent points, they have some more variance in them, i.e. they are scattered. Next, in the clustering-based undersampling technique, we divide the data belonging to each majority class into 10 clusters and choose some portions randomly from each cluster so that the sum of all taken portions has to be equal to the selected threshold that the minority class has been oversampled. Algorithm 2, ADASYN-KM, describes our imbalance processing method.

4.3 Experimental Results and Analysis

We implemented the proposed CNN-based NADS model in Python 3.6.13 and conducted the experiments on a machine with Windows 11 Pro 64 bits operating system.

Algorithm 2 ADASYN-KM

Required:

C_n = The total number of classes;
Training set $S = S_i, i = 1, 2, 3, ...C_n$;
$|S| = N$; #All samples
$cl = 10$ #numbr of clusters

Output:

a balanced training set $S^{'}$;

$Avg = int(N/C_n)$
for $i \leftarrow 1$ to C_n **do**
  if $|Si| < Avg$ **then**
    $S_i^{'} = ADASYN(S_i, Avg)$ # Apply ADASYN on S_i, so that $|S_i^{'}| = Avg$
  end if
  if $|S_i| > Avg$ **then**
    $K_p = K - means(S_i, cl)$ #Making cl clusters from S_i, $p = 1, 2, 3...., cl$
    for $p \leftarrow 1$ to cl **do**
      if $K_p > Avg/cl$ **then**
        $K_p^{'} = Resample(K_p, Avg/cl)$ #Avg/cl random samples from K_p
      else
        $K_p^{'} = Avg/cl$ #assign whole cluster
    end for
    $S_i^{'} = Merge(K_p^{'})$
  end if
  $S^{'} = Merge(S_i^{'})$
end for
return $S^{'}$

The class imbalance process was done on the training set only. The batch size was 256, and the epoch was between 100 to 200.

4.3.i. Classification

The number of convolution kernels and learning rate directly affect classification results in a CNN-based model [74, 75]. We performed experiments on several convolution kernels with different learning rates to obtain a better result. This experiment was on UNSW-NB15 for multiclass classification. We use the "nadam" optimizer and "categorical_crossentropy" loss function for the entire experiment. In our model, the number of convolution kernels in the first four layers of 1D CNN is 64_64_256_256. We use Max Pooling two times to undersample the parameters of the convolution layer. The activation function for the output layer is *softmax*, and for the rest of the layers is *ReLU*. A dropout with a parameter of 0.2 is used after each pooling layer to avoid overfitting. We tested six different number of convolution kernels with seven learning rates. Table 4.3 presents comparative studies of different convolution kernels concerning different sets of learning rates. We observe that the convolution kernels of 64_64_256_256 and learning rate 0.1 outperform in *Recall*, *F-score*, *Train-loss* and *Test-loss* with the score of 97.23%, 97.64%, 0.30%, and 0.07%, respectively. It scores lower by 0.02% in *Accuracy* with a learning rate of 0.01, and *Precision* is lower by 0.02% from 64_64_128_128 with a learning rate of 0.1. The *FAR* is 0.48% which is higher by 0.08% with a learning rate of 0.002. Based on these experiments, we can claim that the convolution kernels 64_64_256_256 outperform the other number of convolution kernels for the majority of the metrics.

Table 4.3: Performance comparison of the proposed model with different number of convolution kernels at different learning-rates for multiclass classification on UNSW-NB15 dataset.

Conv. Kernel No.	Lr	Acc. %	Precision %	Recall %	F-score %	FAR %	Train-time in sec.	Test-time in sec.	Train-loss	Test-loss
16_16_32_32	0.1	98.89	98.28	96.79	97.37	0.06	2876.90	**11.22**	0.34	0.08
	0.03	98.89	98.32	96.85	97.41	0.07	3143.42	13.63	0.34	0.07
	0.01	98.88	98.29	96.82	97.37	0.06	3474.30	14.22	0.34	0.08
	0.008	98.89	98.31	96.94	97.43	0.10	3358.36	13.36	0.35	0.07
	0.006	98.88	98.30	96.85	97.38	0.08	3431.50	15.38	0.35	0.08
	0.004	98.88	98.30	96.92	97.41	0.04	3316.49	16.42	0.34	0.07
	0.002	98.88	98.31	96.86	97.40	0.06	3358.71	17.15	0.34	0.07
16_16_64_64	0.1	98.96	98.29	96.95	97.47	0.48	**2477.50**	45.53	0.33	0.08
	0.03	98.90	98.28	96.77	97.39	0.11	2491.09	46.88	0.33	0.07
	0.01	98.90	98.32	96.60	97.28	0.09	2989.37	55.89	0.33	0.08
	0.008	98.94	98.31	96.84	97.44	0.22	2745.68	63.75	0.33	0.07
	0.006	98.95	98.31	97.03	97.50	0.28	2914.63	44.04	0.33	0.07
	0.004	98.89	98.30	96.82	97.40	0.07	2941.93	63.84	0.33	0.08
	0.002	98.90	98.31	96.84	97.42	0.08	2947.85	63.58	0.33	0.08
32_32_64_64	0.1	98.92	98.37	97.02	97.51	0.16	3225.92	101.26	0.32	0.07
	0.03	98.91	98.38	97.07	97.53	0.14	3293.33	123.06	0.32	0.07
	0.01	98.91	98.38	96.60	97.33	0.12	3417.86	125.12	0.32	0.09
	0.008	98.92	98.37	97.02	97.52	0.16	3351.14	120.91	0.32	0.07
	0.006	98.91	98.34	96.62	97.31	0.08	3287.09	124.21	0.32	0.07
	0.004	98.92	98.37	97.05	97.53	0.15	2955.76	97.15	0.32	0.07
	0.002	98.91	98.37	97.02	97.51	0.13	3291.78	127.17	0.32	0.07
32_32_128_128	0.1	98.94	98.37	97.08	97.55	0.26	4086.70	215.18	0.31	0.07
	0.03	98.99	98.37	97.03	97.55	0.41	4183.27	230.20	0.31	0.07
	0.01	98.91	98.37	96.97	97.49	0.14	3801.17	196.05	0.31	0.07
	0.008	98.92	98.37	97.02	97.52	0.13	4214.31	223.80	0.31	0.07
	0.006	98.92	98.39	96.99	97.51	0.10	4217.15	224.74	0.31	0.07
	0.004	98.93	98.40	96.94	97.51	0.13	4135.18	218.18	0.31	0.07
	0.002	98.93	98.37	96.86	97.45	0.17	4172.41	229.15	0.31	0.07
64_64_128_128	0.1	98.96	**98.40**	97.14	97.59	0.29	5237.70	326.10	0.30	0.07
	0.03	98.98	98.39	97.10	97.58	0.29	4874.65	275.23	0.30	0.07
	0.01	98.99	98.38	97.19	97.61	0.39	5482.30	348.37	0.30	0.07
	0.008	98.99	98.39	97.12	97.60	0.35	5437.98	415.15	0.30	0.07
	0.006	98.98	98.38	97.17	97.60	0.38	5627.75	420.98	0.30	0.07
	0.004	98.97	98.38	97.16	97.59	0.29	5372.79	364.23	0.30	0.07
	0.002	98.96	98.39	97.15	97.59	0.29	5496.99	374.56	0.30	0.07
64_64_256_256	0.1	99.01	98.38	**97.23**	**97.64**	0.48	6029.59	339.09	**0.30**	**0.07**
	0.03	98.99	98.39	97.14	97.60	0.37	6672.53	389.28	0.30	0.07
	0.01	**99.03**	98.38	97.03	97.57	0.48	6949.32	414.00	0.30	0.07
	0.008	99.02	98.37	97.14	97.61	0.49	7373.04	439.24	0.30	0.07
	0.006	98.99	98.37	97.17	97.61	0.40	7387.88	440.12	0.30	0.07
	0.004	99.00	98.38	97.09	97.59	0.42	7620.89	454.00	0.30	0.07
	0.002	99.00	98.38	97.16	97.61	**0.40**	7846.29	467.43	0.30	0.07

To evaluate the effectiveness of our model on different datasets and different types of classifications, we implemented the same experiment on binary classification and multiclass classification on both datasets. Table 4.4 provides a summary results for all the metrics on both datasets.

Table 4.5 presents per class performance of the model on UNSW-NB15 dataset. we can see that the detection rate for the minority classes of Shellcode and Worms is better concerning the deficient number of samples for the classes.

4.3.ii. Discussion

The experimental results show that our proposed method, combining an imbalance processing technique and a 1D CNN-based classifier model, significantly improved the detection rate and reduced the FAR. The main reason is that instead of random resampling, we employed a combined method of ADASYN oversampling and a cluster-based undersampling using the K-means algorithm. We set the number of clusters manually to 10 due to better results from a small

Table 4.4: Overall performance of the proposed model in binary and multiclass classification on UNSW-NB15 and NSL-KDD datasets.

Dataset	**UNSW-NB15**		**NSL-KDD**	
Classification	**Binary**	**Multiclass**	**Binary**	**Multiclass**
Accuracy %	98.79	99.01	98.52	98.97
Precision %	98.90	98.38	98.52	98.35
Recall %	98.79	97.23	98.52	98.16
F-score %	98.81	97.64	98.52	98.22
FAR %	0.18	0.48	1.48	0.48
Train-time in sec.	8049.91	6029.59	641.41	626.41
Test-time in sec.	388.31	339.09	15.67	14.90
Train_loss	0.03	0.30	0.09	0.17
Test_loss	0.02	0.07	0.05	0.07

Table 4.5: Class-wise performance evaluation of the proposed model on the UNSW-NB15 dataset.

Class	**Accuracy %**	**Precision %**	**Recall %**	**F-score %**	**Support**
Normal	98.92	99.92	98.84	99.38	436,755
Generic	99.83	99.84	98.21	99.02	42,418
Exploits	98.97	84.07	50.54	63.13	8,764
Fuzzers	98.90	45.22	69.70	54.85	4,773
DoS	98.90	33.48	71.74	45.65	3,220
Reconnaissance	99.82	85.76	80.28	82.93	2,753
Analysis	99.69	11.34	28.46	16.22	527
Backdoor	99.77	10.82	20.74	14.22	458
Shellcode	99.68	13.13	78.52	22.50	298
Worms	99.97	16.13	73.53	26.46	34
Overall/avg.	**99.01**	**98.38**	**97.23**	**97.64**	**500,000**

pre-test. Cluster-based undersampling prevents information loss, which may occur by eliminating samples randomly. In this way, it will contribute to good performance. On the other hand, ADASYN oversampling generates synthetic points which are more realistic than the randomly generated points. To demonstrate the effectiveness of our model, we compared our results with some of the previous works given in Table 4.6. We use *Accuracy*, *Precision*, *Recall*, *f-measure*, and *FAR* metrics for the comparison and can see from the table that our model performs much higher than the previous works. Reducing the false positive rate (false alarm rate) is one of the key challenges in network anomaly detection, which is significantly dropped by our method.

Table 4.6: Comparison of multiclass classification results with some of the previous works.

Dataset	**Classification**	**Model**	**Acc. %**	**Precision%**	**DR %**	**F-score %**	**FAR %**
UNSW-NB15	Multiclass	CSCADE-ANN [76]	86.40	86.74	93.38	89.94	-
		ICVAE-DNN [77]	89.08	86.05	95.68	90.61	19.01
		In this work	**99.01**	**98.38**	**97.23**	**97.64**	**0.48**
NSL-KDD	Multiclass	SCDNN [78]	72.64	-	57.48	-	-
		RNN-IDS [30]	81.29	-	-	-	-
		ID-CVAE [79]	80.10	81.59	80.10	79.08	-
		Gaussian–Bernoulli RBM [80]	73.23	62.33	95.09	75.30	-
		ICVAE-DNN [77]	85.97	97.39	77.43	86.27	2.74
		In this work	**98.97**	**98.35**	**98.16**	**98.22**	**0.48**

4.4 Conclusion

We proposed a combined approach to process class imbalance to overcome the class imbalance issue in network traffic data. The technique combines ADASYN oversampling and clustering-

based undersampling using the K-means algorithm. We develop a 1D CNN-based network anomaly detection with 64_64_256_256 number of convolutional kernels, a learning rate of 0.1, a softmax activation function for the output layer, ReLU activation function for the layers other than the output layer, and Max Pooling followed by a dropout. We evaluated our model on two commonly used datasets, UNSW-NB15 and NSL-KDD. The binary and multiclass classification was conducted on both datasets. A comparative study with six different convolution kernels and seven learning rates was conducted to show how our model performs vs the other convolution kernels. The experimental results show that the model produces a detection rate of 98.79% and 97.23% on UNSW-NB15 for binary classification and multiclass classification, respectively, which is the highest among all tested scenarios. In the evaluation using NSL-KDD, the model yields a detection rate of 98.52% for binary type classification and 98.16% for multiclass type classification. We compared our method with some previous work on multiclass classification and observed that our model performs superior to state-of-the-art models and points to a promising direction for future network anomaly detection with large scale and imbalanced datasets.

Chapter 5

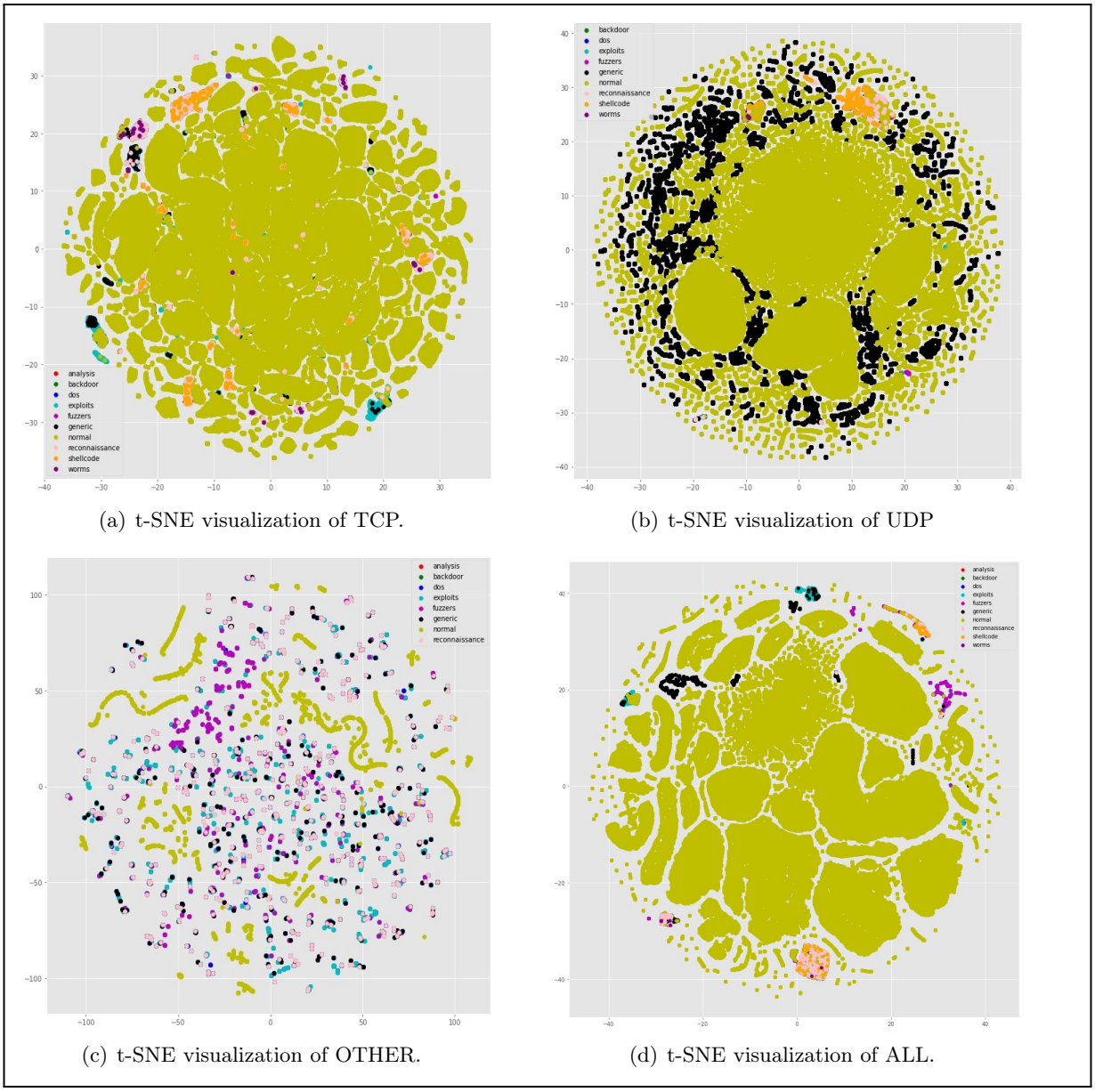
Protocol-wise Traffic Analysis for Network Anomaly Detection [1]

5.1 Introduction

In this study, 1D CNN architecture is trained on samples of UNSW-NB15 and NSL-KDD datasets. From feature visualization of the UNSW-NB15 in Figure 5.1 and similar visualization for NSL-KDD, we observe that the data distribution is different from protocol to protocol, the overlap of classes and distribution is completely different. This difference will impact on classifier's behaviour, so we split the data into different protocol categories first, and then treat each one separately. On the other hand, from Tables 5.1 and 5.2, we can see that the datasets are very imbalanced; for example, 'Worms', 'Shellcode', 'Backdoor', and 'Analysis' are very less in terms of the number of instances compared to the other classes in all protocol categories in the UNSW-NB15; Similarly, R2L and U2R have fewer samples in the NSL-KDD. In addition to that, some classes do not exist under some protocol category; for example, 'Analysis' does not exist under UDP, and 'Worms' and 'Shellcode' do not exist under the 'OTHER' category in the UNSW-NB15 dataset; likewise, U2R doesn't exist under the 'OTHER' category in the NSL-KDD. To tackle the issues of imbalance; first, we randomly undersampled the majority classes and then applied SMOTE oversampling technique to minority classes for each protocol category separately; we used a mix of undersampling and oversampling to minimize each one's drawbacks. To mitigate the risk of overlap problem, we treat data in a protocol-wise manner. The main steps which we followed in this study are data preprocessing, data division into protocols, feature selection using the Chi-square technique, sampling, and applying 1D CNN for multiclass classification.
Overall, this study has made the following contributions in the domain of network anomaly detection:

- Features visualization is conducted using the t-SNE method prior to training the model. From Figures 5.1, we observed that the distribution of different classes varies from protocol to protocol, so we divided the data into different protocol categories and then performed classification on each category as well as on the total data together. We compared the experimental results on the data treated based on separate protocols with the results of the data treated collectively and observed that the model is performing better in protocol wise multiclass classification compared to all protocols combined together.

[1] Mohammad Kazim Hooshmand and Doreswamy. **Network Anomaly Detection Using Deep Learning Techniques**. *CAAI Transactions on Intelligence Technology*. 2022. Vol. 7, Issue 2, pp. 228-243, ISSN: 2468-2322.



(a) t-SNE visualization of TCP.

(b) t-SNE visualization of UDP

(c) t-SNE visualization of OTHER.

(d) t-SNE visualization of ALL.

Figure 5.1: t-SNE visualization of UNSW-NB15 dataset.

- An integrated 1D CNN based network anomaly detection model was developed to fuse dimensionality reduction, imbalance processing technique, overlap problem handling, and classification.

5.2 The Proposed Method

The main steps of our proposed method are 1) data preprocessing, 2) dividing data into protocol categories such as TCP, UDP, OTHER, and ALL, 3) random undersampling of the majority classes and oversampling of the minority classes, and 4) 1D CNN based classification. Figure 5.2 describes the architecture of our method.

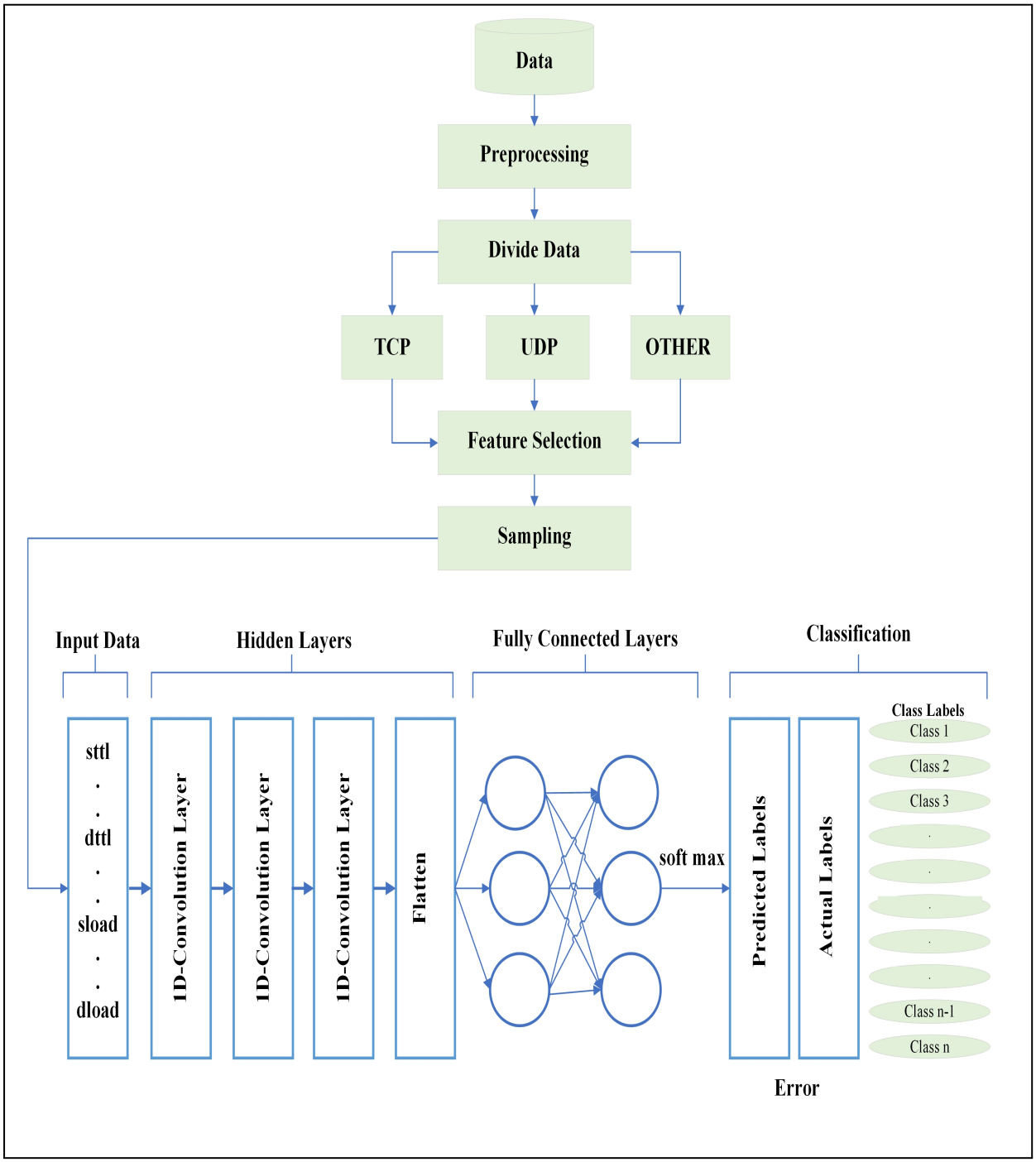


Figure 5.2: Architecture of the proposed model.

5.2.i. Data Preprocessing

Tables 5.7 and 5.8 give the distribution of classes in different protocols in the sample data of both the datasets which we use in this chapter. Redundant features such as switch information ('srcip', 'sport', 'dstip', 'dsport') and time-related features ('ltime' and 'stime') are eliminated from the dataset. Categorical features on both datasets have many values. We apply the one-hot encoding technique to encode these features, which will add new variables for each value in each categorical feature. The process will be very complex because it will add dozens of new features to the data. To avoid the complexity, prior to encoding, we categorize categorical values, so that

feature values with more instances are kept as it is, and the remaining feature values with fewer instances are put collectively under the 'OTHER' category. Reference to Tables 5.3 and 5.4, we decided which feature value has to be kept as it is and which ones have to be put under the 'OTHER' category. For example, in the UNSW-NB15, under the 'proto' categorical feature, we kept TCP and UDP as it is because both contribute to about 98% of the total instances, and the remaining, which contribute about 2%, are grouped under the 'OTHER'. The categorization for the UNSW-NB15 is as follows:

Table 5.1: Distribution of classes over different protocols in the UNSW-NB15.

No	Class	TCP	UDP	OTHERS	All
1	**Normal**	1,436,890	766,685	15,186	2,218,761
2	**Generic**	3,118	210,600	1,763	215,481
3	**Exploits**	27,443	874	16,208	44,525
4	**Fuzzers**	15,474	6,043	2,729	24,246
5	**DoS**	3,336	527	12,490	16,353
6	**Reconnaissance**	6,965	4,890	2,132	13,987
7	**Analysis**	622	-	2,055	2,677
8	**Backdoor**	323	34	1,972	2,329
9	**Shellcode**	750	761	-	1,511
10	**Worms**	153	21	-	174
	Total	**1,495,074**	**990,435**	**54,535**	**2,540,044**

Table 5.2: Distribution of classes over different protocols in NSL-KDD.

No	Class	TCP	UDP	OTHERS	All
1	**Normal**	61,442	14,210	1,402	77,054
2	**DoD**	48,926	906	3,553	53,385
3	**Probe**	7,721	1,981	4,375	14,077
4	**R2L**	3,231	514	4	3,749
5	**U2R**	249	3	-	252
	Total	**1,21,569**	**17,614**	**9,334**	**1,48,517**

- **Proto:** 'TCP', 'UDP', and 'OTHER'
- **State:** 'FIN', 'CON', 'INT', and 'OTHER'
- **Service:** '-', 'DNS', and 'OTHER'

Similarly, the same method is applied for the other categorical features on the UNSW-NB15 as well as on the NSL-KDD datasets. The categorisation for the NSL-KDD is as follows:

- **Protocol_type:** 'TCP', 'UDP', and 'OTHER'
- **Service:** 'HTTP', 'PRIVATE', 'DOMAIN_U', 'SMTP', 'FTP_DATA', 'ECO_I', 'TELNET', 'ECR_I', 'FTP', 'FINGER', 'AUTH', 'IMAP4', 'Z39_50', 'UUCP', 'COURIER', 'BGP', 'ISO_TSAP', 'UUCP_PATH', 'WHOIS', 'TIME', 'NNSP', 'VMNET', 'URP_I', 'DOMAIN', 'CTF', and 'OTHER',
- **Flag:** 'SF', 'S0', 'REJ', 'RSTR', 'RSTO', and 'OTHER',

There are few null values and spelling differences exist in columns 'ct_ftp_cmd', 'is_ftp_login', and 'ct_flw_http_mthd' in the UNSW-NB15, which are handled appropriately, and all null

values under 'attack_cat' feature are replaced with 'Normal'. The nominal features are encoded, and then the normalization is applied using the min_Max scaling technique. Equation 5.1 describes the way in which the min_Max method transforms values in the range of 0 and 1.

Table 5.3: Categorical features of UNSW-NB15.

Categorical Features	Type	Percentage
Proto	TCP	58.9%
	UDP	39.0%
	OTHER	2.1%
State	FIN	58.2%
	CON	22.1%
	INT	19.3%
	OTHER	0.4%
Service	-	49.1%
	DNS	30.8%
	OTHER	20.2%

Table 5.4: Categorical features of NSL-KDD.

Categorical Features	Type	Percentage
Protocol_type	TCP	81.9%
	UDP	11.9%
	OTHER	6.3%
Service	HTTP	32.4%
	PRIVATE	17.9%
	DOMAIN_U	6.7%
	SMTP	5.6%
	FTP_DATA	5.2%
	ECO_I	3.3%
	TELNET	2.7%
	ECR_I	2.6%
	FTP	1.6%
	FINGER	1.3%
	AUTH	0.7%
	IMAP4	0.6%
	Z39_50	0.6%
	UUCP	0.6%
	COURIER	0.5%
	BGP	0.5%
	ISO_TSAP	0.5%
	UUCP_PATH	0.5%
	WHOIS	0.5%
	TIME	0.5%
	NNSP	0.5%
	VMNET	0.4%
	URP_I	0.4%
	DOMAIN	0.4%
	CTF	0.4%
	OTHER	13.1%
Flag	SF	60.5%
	S0	24.8%
	REJ	10.2%
	RSTR	2.1%
	RSTO	1.6%
	OTHER	0.9%

$$X = \frac{X_i - X_{minimum}}{X_{maximum} - X_{minimum}} \tag{5.1}$$

where X stands for the normalized data, X_i represents the original feature value, $X_{minimum}$ is the minimum value in the feature, and $X_{maximum}$ is the maximum value in the feature before the normalization.

5.2.ii. Data Division

From the feature visualization given in Figures 5.1 and similar feature visualization for NSL-KDD, we observed that both datasets have an overlap problem. The overlap problem is that

one class mimics another class's behaviours [81]. If a model is trained without handling the overlap problem, the outcome will not be satisfactory [81]. So to reduce the impact of the overlap problem, prior to feature selection, we split the datasets based on different protocols. As discussed in 5.2.i., we keep protocols with a large number of instances as it is and categorize the remaining as 'OTHER'. Tables 5.3 and 5.4 show which protocols are kept as it is and which are put under the 'OTHER'. So we split both datasets into three parts and treat each part as independent data hereafter. The different protocol categories have gone through the independent processes from feature selection to classification because the different protocol formats require a different set of features subsequently.

5.2.iii. Feature Selection

Feature selection is necessary to eliminate less essential features from a dataset. We reduced the number of features of all categories (TCP, UDP, OTHER, and ALL) to almost 50% using the Chi-square selection technique. The 'ALL' category is the combination of TCP, UDP, and OTHER after resampling. Undersampling and oversampling are explained in 5.2.iv..

Table 5.5: Selected features of each categories on UNSW-NB15.

Category	Selected Features
TCP	dur, sttl, dttl, sload, dload, smeansz, dmeansz, sjit, sintpkt, dintpkt, tcprtt, synack, ackdat, ct_flw_http_mthd, ct_state_ttl, ct_srv_dst, ct_srv_src, ct_src_ltm, ct_dst_ltm, is_ftp_login, service
UDP	dur, sttl, sbytes, sload, dttl, dload, dpkts, spkts, smeansz, sjit, dmeansz, sintpkt, ct_src_dport_ltm, ct_state_ttl, ct_dst_ltm, state, ct_dst_src_ltm, service
OTHER	sbytes, sttl, sload, dpkts, smeansz, dmeansz, sjit, djit, sintpkt, dintpkt, ct_state_ttl, ct_srv_src, ct_srv_dst, ct_src_ltm, ct_dst_ltm, ct_dst_src_ltm, ct_dst_sport_ltm, is_sm_ips_ports, ct_src_dport_ltm, state
ALL	sttl, dload, dttl, sload, smeansz, sintpkt, dmeansz, dintpkt, tcprtt, ackdat, synack, ct_state_ttl, ct_srv_src, ct_dst_ltm, ct_srv_dst, is_sm_ips_ports, proto, state, service

Tables 5.5 and 5.6 show lists of selected features using the Chi-square selection technique for different categories on the UNSW-NB15 and NSL-KDD, respectively.

5.2.iv. Sampling

Recently, different methods have been proposed to solve the issue of class imbalance in a dataset. The methods are broadly grouped under two categories: algorithm level and data level [82]. The algorithm level approach works on modifying ML algorithms to accommodate an imbalanced dataset. Examples of such modifications can be cost-sensitive methods/cost error minimization rather than accuracy rate maximization [83]. The data level techniques are trying to re-balance the data class distribution either by decreasing the majority class or increasing the minority class. Various sampling methods exist; some of the examples can be random undersampling, random oversampling, direct oversampling, SMOTE, and many others [82]. Oversampling techniques add some more instances to the minority class in the training set, the simplest method is random oversampling [84], but the drawback is overfitting [85]. To overcome this problem, Chawla [86] proposed SMOTE technique, which generates synthetic samples from the minority class. Samples created by SMOTE technique are a linear combination of two identical samples from the minority class [40, 84, 86]. SMOTE oversampling algorithm works as follows: Let S represent size of a small class, considering a sample j of the small size class, and x_j denotes its feature vector such that, $j \epsilon \{1, ..., S\}$:

Table 5.6: Selected features of each categories on NSL-KDD.

Category	Selected Features
TCP	duration, count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate, logged_in, is_guest_login, service_ftp, service_http, service_imap4, service_other, service_private, service_smtp, service_telnet, service_uucp, flag_rej, flag_rsto, flag_rstr, flag_s0, flag_sf
UDP	duration, src_bytes, dst_bytes, wrong_fragment, count, srv_count, serror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_rerror_rate, service_domain_u, service_other, service_private
OTHER	src_bytes, wrong_fragment, count, srv_count, serror_rate, rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_rerror_rate, service_eco_i, service_ecr_i, service_other, service_urp_i, flag_sf
ALL	wrong_fragment, count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate, logged_in, proto_icmp, proto_udp, service_domain_u, service_eco_i, service_ecr_i, service_ftp, service_http, service_imap4, service_other, service_private, service_smtp, service_telnet, service_urp_i, flag_rej, flag_rsto, flag_rstr, flag_s0, flag_sf

- Find k neighbours of the sample x_j from all S (using Euclidean Distance for example) and denoted it as $x_{j\,(near)}$, $near\,\epsilon\,\{1, ..., k\}$.
- $x_{j\,(nn)}$ sample is selected randomly from the k neighbours, and random number β_1 between 0 and 1 is generated to synthesize a new sample x_{j1} as in the equation $x_{j1} = x_j + \beta_1 * (x_{j\,(nn)} - x_j)$.
- Step 2 is repeated M times to synthesize M new samples: $x_{j\,(new)}$, $new\,\epsilon\,\{1, ..., M\}$.

In our case, we are undersampling the majority class using random undersampling. Then after feature selection, we applied SMOTE oversampling approach to oversample the minority classes in the training set. Table 5.7 provides the size of each class sample concerning protocol categories after random undersampling in the UNSW-NB15. Similarly, Table 5.8 shows the sample distribution for the NSL-KDD.

Figure 5.2 shows all main steps of our proposed method, the CNN model contains three 1-D layers with *ReLU* activation function followed by a flatten layer. Classification is done using two fully connected layers using *softmax* activation.

5.3 Experimental Results

The hardware and software configuration for our experiments are Windows 10 with an Intel®Core™ i7-9700 CPU running at 3.00 GHz, 8 processors, 16 GB RAM, and 1 TB hard disk. The python 3.6.13 with libraries of *scikit learn* and *Keras* [87, 88] was utilized. As CNN is a parameterized function, its performance is fully dependent on the optimal parameters; some of the parameters are the number of epochs, number of hidden layers, number of nodes in each layer etc.

Table 5.7: Sample size of each protocol from UNSW-NB15.

Class	TCP	UDP	OTHER	ALL
Normal	21,553	23,001	9,113	53,667
Fuzzers	5,014	5,004	2,729	12,747
Reconnaissance	4,855	4,890	2,132	11,877
Exploits	5,214	874	5,024	11,112
Generic	3,118	5,054	1,763	9,935
DoS	3,336	527	4,746	8,609
Analysis	622	-	2,055	2,677
Backdoor	323	34	1,972	2,329
Shellcode	750	761	-	1511
Worms	153	21	-	174
Total	44,938	40,166	29,534	114,638

Table 5.8: Sample size of each protocol from NSL-KDD.

Class	TCP	UDP	OTHER	ALL
Normal	10,445	6,252	1,402	18,099
DoS	10,274	906	3,553	14,733
Probe	7,721	1,981	4,375	14,077
R2L	3,231	514	4	3,749
U2R	249	3	-	252
Total	31,920	9,656	9,334	50,910

In [35], authors claim that lower learning rates have shown better results in differentiating the connection records using CNN. It is also mentioned that to attain an acceptable detection rate, a lower learning rate anticipates more number of epochs for lower frequency classes. In [89], the learning rate was assigned to 0.01 and *adam* optimizer was utilized for CNN architecture. In [90], authors claim that, in the case of multiclass classification, *ReLU* activation function was performing better than *sigmoid* and *tanh* for 500 epochs. We have conducted different parameters adjustment in tuning phase. The number of epochs was tested for 100-500, we decided 500 because of its higher performance for all the categories of data (TCP, UDP, OTHER). *optimizer* is set to 'adam' with default parameters, *loss* is set to 'categorical_crossentropy', and *metrics* is set to *accuracy*, batch size is 1, number of nodes in each layer is 512, and activation function is *ReLU*. The required time for running our model on the given configuration is calculated as follows: by calculating the required time for a single epoch execution and the number of epochs, we found that it takes approximately 7s to run one epoch. In the proposed method, we have considered 500 epochs, so the total time complexity of the model is 3500s. As per big O notations with respect to the number of loops and iterative approach, the complexity is $O(n)$. The metrics we used are: *Accuracy*, *Precision*, *Recall*, and *F-score*. We used accuracy measure to see the overall prediction of our model, it is good if you have a systematic dataset (false negatives false positives counts are close). Since our datasets have uneven class distribution, Accuracy will not give an accurate idea of every class, so we have used the F-score, which is best for uneven class data. Precision is used to be confident of our actual positive value, and Recall is used to get an understanding on the sensitivity of our prediction.

Tables 5.9 - 5.13 show confusion matrices for the UNSW-NB15 dataset, while Tables 5.14 - 5.18 represent confusion matrices for NSL-KDD dataset.

Table 5.9 represents the confusion matrix of 'TCP'; Table 5.10 shows the confusion matrix of 'UDP'; Table 5.11 provides the confusion matrix of 'OTHER'; similarly, Tables 5.12 and 5.13

Table 5.9: TCP Confusion matrix for UNSW-NB15.

Act.pred.	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode	Worms
Analysis	176 94%	0	9	2	0	0	0	0	0	0
Backdoor	15	52 54%	4	0	2	10	0	6	6	2
DoS	254	5	322 32%	284	30	45	1	19	33	8
Exploits	284	8	90	947 61%	31	62	1	97	25	19
Fuzzers	71	0	7	3	1390 92%	5	0	10	16	2
Generic	228	8	28	63	34	557 60%	1	4	9	3
Normal	15	0	1	1	55	0	6391 99%	0	3	0
Reconnaissance	66	2	8	10	8	2	0	1338 92%	19	4
Shellcode	35	0	0	0	21	0	0	7	162 72%	0
Worms	7	1	2	4	0	0	0	0	0	32 70%

Table 5.10: UDP Confusion matrix for UNSW-NB15.

Act.Pred.	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode	Worms
Backdoor	10 100%	0	0	0	0	0	0	0	0
DoS	20	78 49%	30	6	0	0	11	8	5
Exploits	25	26	177 68%	14	1	0	0	16	3
Fuzzers	39	17	58	1358 90%	0	0	2	19	8
Generic	1	3	1	0	1510 100%	0	0	0	1
Normal	1	4	1	62	0	6833 99%	0	0	0
Reconnaissance	3	2	0	0	0	0	1460 100%	2	0
Shellcode	38	4	2	7	0	0	6	170 74%	2
Worms	0	0	1	0	0	0	0	0	5 83%

Table 5.11: OTHER Confusion matrix for UNSW-NB15.

Act.Pred.	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance
Analysis	300 49%	90	9	47	153	12	0	5
Backdoor	25	436 67%	6	17	70	60	0	32
DoS	17	9	1296 91%	33	0	21	0	48
Exploits	11	43	28	1395 93%	2	23	0	5
Fuzzers	46	16	1	3	743 91%	0	0	10
Generic	44	56	45	71	7	280 53%	0	22
Normal	0	0	0	0	0	0	2733 100%	1
Reconnaissance	39	27	23	42	10	50	0	449 70%

Table 5.12: ALL Confusion matrix for UNSW-NB15.

Act. Pred.	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode	Worms
Analysis	795 99%	0	7	0	1	0	0	0	0	0
Backdoor	593	84 12%	2	0	3	2	0	5	10	0
DoS	1801	18	272 11%	242	44	59	3	32	88	24
Exploits	1934	20	55	1000 30%	55	76	2	73	70	49
Fuzzers	986	3	0	14	2657 69%	2	1	19	122	20
Generic	735	26	32	50	37	2059 69%	1	6	28	7
Normal	43	0	1	1	109	0	15938 99%	1	7	0
Reconnaissance	750	3	7	3	8	0	0	2749 77%	26	17
Shellcode	18	0	0	0	32	0	0	17	385 85%	1
Worms	7	1	0	0	1	2	0	0	1	40 77%

Table 5.13: Computed All Confusion matrix for UNSW-NB15.

Act.Pred.	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode	Worms
Analysis	476 59%	90	18	49	153	12	0	5	0	0
Backdoor	40	498 66%	10	17	72	70	0	38	6	2
DoS	271	34	1696 66%	347	36	66	1	78	41	13
Exploits	295	76	144	2519 76%	47	86	1	102	41	22
Fuzzers	117	55	25	64	3491 91%	5	0	22	35	10
Generic	272	65	76	135	41	2347 79%	1	26	9	4
Normal	15	1	5	2	117	0	15957 99%	1	3	0
Reconnaissance	105	32	33	52	18	52	0	3247 91%	21	4
Shellcode	35	38	4	2	28	0	0	13	334 73%	2
Worms	7	1	2	5	0	0	0	0	0	37 71%

Table 5.14: TCP Confusion matrix for NSL-KDD.

Act.Pred.	DoS	Probe	R2L	U2R	Normal
DoS	3073 100%	4	0	0	5
Probe	1	2300 99%	3	3	9
R2L	0	3	926 96%	23	17
U2R	0	0	6	65 87%	4
Normal	17	4	70	15	3028 97%

Table 5.15: UDP Confusion matrix for NSL-KDD.

Act.Pred.	DoS	Probe	R2L	U2R	Normal
DoS	270 99%	0	0	0	2
Probe	0	587 99%	2	2	3
R2L	0	1	150 97%	1	2
U2R	0	0	0	1 100%	0
Normal	2	5	52	0	1817 97%

Table 5.16: OTHER Confusion matrix for NSL-KDD.

Act.Pred.	DoS	Probe	R2L	Normal
DoS	1065 100%	0	0	1
Probe	0	1307 100%	0	6
R2L	0	0	1 100%	0
Normal	4	9	0	408 97%

Table 5.17: ALL Confusion matrix for NSL-KDD.

Act.Pred.	DoS	Probe	R2L	U2R	Normal
DoS	4406 100%	0	0	0	14
Probe	4	4194 99%	7	1	17
R2L	4	2	1082 96%	16	21
U2R	1	0	7	64 85%	3
Normal	19	61	132	36	5182 95%

Table 5.18: Computed All Confusion matrix for NSL-KDD.

Act.Pred.	DoS	Probe	R2L	U2R	Normal
DoS	4408 100%	4	0	0	8
Probe	1	4194 99%	5	5	18
R2L	0	4	1077 96%	24	19
U2R	0	0	6	66 87%	4
Normal	23	18	122	15	5253 97%

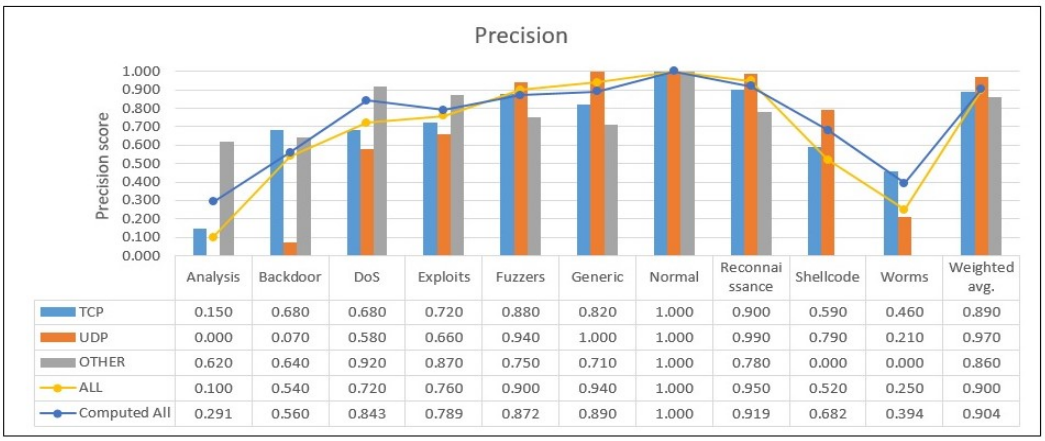
provide the confusion matrices of 'ALL' and 'Computed All', respectively, using UNSW-NB15. Likewise, for the NSL-KDD; Table 5.14 represents the confusion matrix of 'TCP'; Table 5.15 shows the confusion matrix of 'UDP'; Table 5.16 provides the confusion matrix of 'OTHER'; similarly, Tables 5.17 and 5.18 provide the confusion matrices of 'ALL' and 'Computed All'. Here, 'TCP', 'UDP', and 'OTHER' are independent protocol data. 'ALL' is the combination of all the three categories treated collectively without splitting, and 'Computed All' is the combination of all categories treated protocol-wise, and the result is calculated later. The diagonal red highlighted percentages show the normalized detection rate in each confusion matrix.

By comparing Tables 5.12 and 5.13, we observe that Computed All is scoring better for the majority of the classes except for Normal, which is equal and for 'Analysis', 'Shellcode', and 'Worms' which is lower. Also, by comparing Tables 5.17 and 5.18, we observe that Computed All is scoring better for Normal and U2R classes.

5.4 Results Analysis and Comparison

5.4.i. Comparison

Figure 5.3 provides *Precision* score comparison for all categories. Figures 5.4 and 5.5 show the comparison of *Recall* and *F-score*, respectively, using UNSW-NB15. Figure 5.6 gives *Precision* score comparison for all categories. Figures 5.7 and 5.8 show the comparison of *Recall* and *F-score*, respectively, using NSL-KDD. We can see that 'Computed All' scores better than 'ALL' in terms of Precision, Recall, and F-score weighted average on both datasets. It shows that treating protocol-wise data results in a better performance.

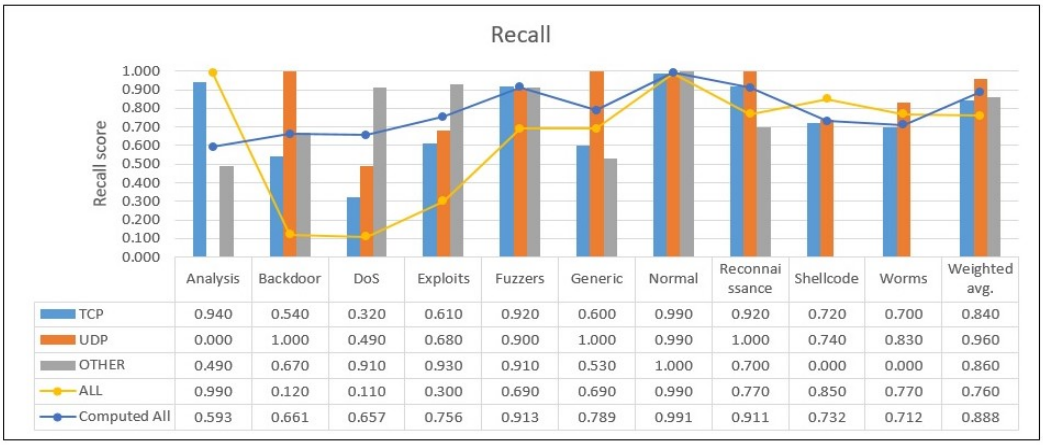


	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode	Worms	Weighted avg.
TCP	0.150	0.680	0.680	0.720	0.880	0.820	1.000	0.900	0.590	0.460	0.890
UDP	0.000	0.070	0.580	0.660	0.940	1.000	1.000	0.990	0.790	0.210	0.970
OTHER	0.620	0.640	0.920	0.870	0.750	0.710	1.000	0.780	0.000	0.000	0.860
ALL	0.100	0.540	0.720	0.760	0.900	0.940	1.000	0.950	0.520	0.250	0.900
Computed All	0.291	0.560	0.843	0.789	0.872	0.890	1.000	0.919	0.682	0.394	0.904

Figure 5.3: Precision comparison UNSW-NB15.

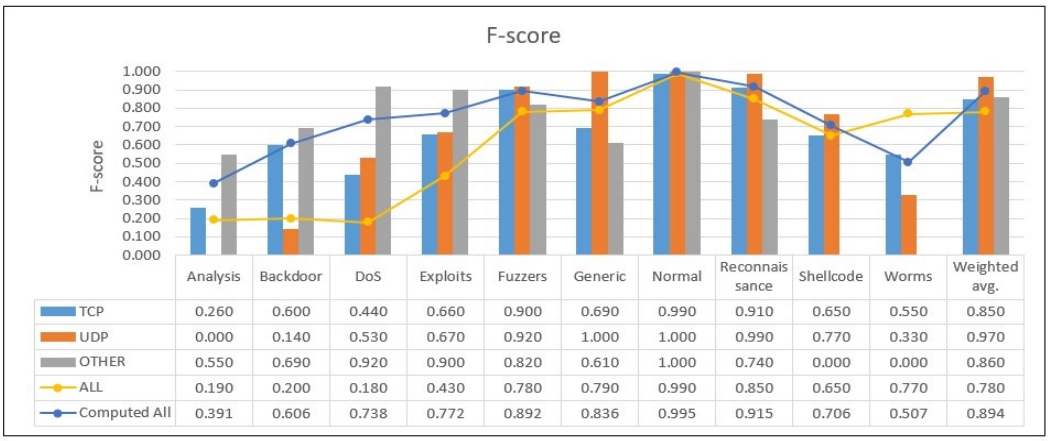
To demonstrate our method's effectiveness, we have compared our experimental results with some previous works. Table 5.19 shows an overall multiclass classification performance compared with some previous works. We can see that using the NSL-KDD, our method outperforms in all the metrics considered for the comparison, while using the UNSW-NB15 dataset, it scores better than all for the Precision, but in terms of Recall and F-score, it yields lower from the work [77] and higher than the others.

We also compared the detection rate score for each class. Table 5.20 shows a class-wise detection rate comparison between our method and two previous works. We observe that our method outperforms in all classes using the NSL-KDD datasets. For the UNSW-NB15, our approach



	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode	Worms	Weighted avg.
TCP	0.940	0.540	0.320	0.610	0.920	0.600	0.990	0.920	0.720	0.700	0.840
UDP	0.000	1.000	0.490	0.680	0.900	1.000	0.990	1.000	0.740	0.830	0.960
OTHER	0.490	0.670	0.910	0.930	0.910	0.530	1.000	0.700	0.000	0.000	0.860
ALL	0.990	0.120	0.110	0.300	0.690	0.690	0.990	0.770	0.850	0.770	0.760
Computed All	0.593	0.661	0.657	0.756	0.913	0.789	0.991	0.911	0.732	0.712	0.888

Figure 5.4: Recall comparison UNSW-NB15.



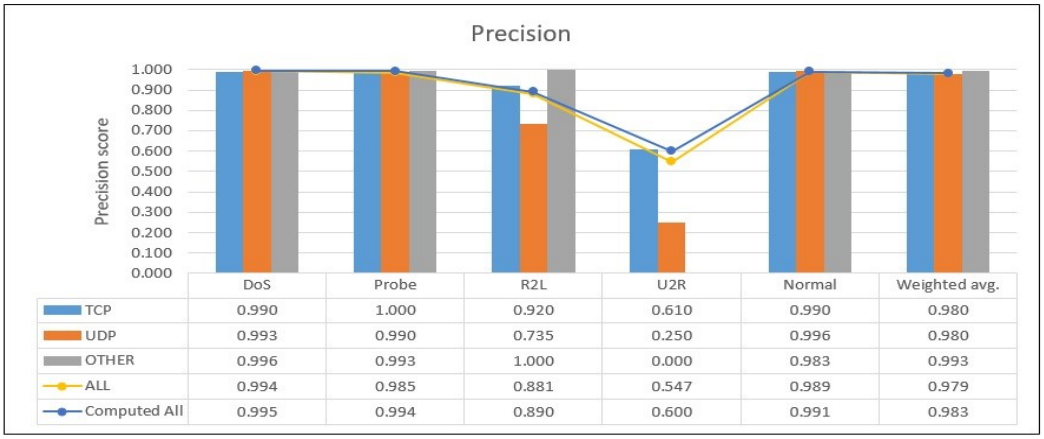
	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode	Worms	Weighted avg.
TCP	0.260	0.600	0.440	0.660	0.900	0.690	0.990	0.910	0.650	0.550	0.850
UDP	0.000	0.140	0.530	0.670	0.920	1.000	1.000	0.990	0.770	0.330	0.970
OTHER	0.550	0.690	0.920	0.900	0.820	0.610	1.000	0.740	0.000	0.000	0.860
ALL	0.190	0.200	0.180	0.430	0.780	0.790	0.990	0.850	0.650	0.770	0.780
Computed All	0.391	0.606	0.738	0.772	0.892	0.836	0.995	0.915	0.706	0.507	0.894

Figure 5.5: F-score comparison UNSW-NB15.

Table 5.19: Overall performance comparison with previous works.

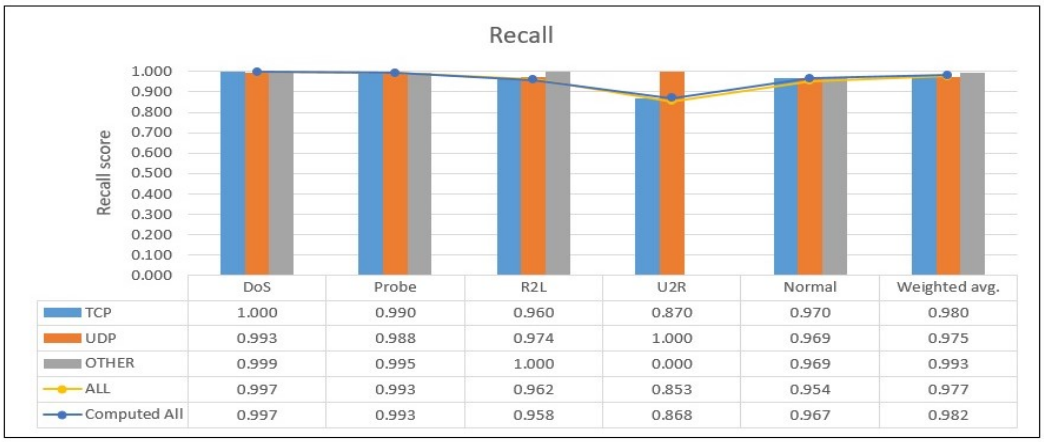
Dataset	Method	Precision	Recall	F-score
UNSW-NB15	Vinayakumar Ravi et al. [90]	0.60	0.65	0.59
	Yang et al. [77]	0.86	**0.96**	**0.91**
	Al-Turaiki et al. [10]	0.84	0.69	0.74
	Our Method (Computed All)	**0.90**	0.89	0.89
NSL-KDD	Vinayakumar Ravi et al. [90]	0.81	0.79	0.77
	Yang et al. [77]	0.97	0.77	0.86
	Al-Turaiki et al. [10]	0.83	0.81	0.80
	Our Method (Computed All)	**0.98**	**0.98**	**0.98**

scores higher than the work [90] in the majority of attacks, except for 'DoS'. Similarly, our method yields better results for the majority of classes compared to [77], except for 'Generic', 'Shellcode', and 'Worms'.



	DoS	Probe	R2L	U2R	Normal	Weighted avg.
TCP	0.990	1.000	0.920	0.610	0.990	0.980
UDP	0.993	0.990	0.735	0.250	0.996	0.980
OTHER	0.996	0.993	1.000	0.000	0.983	0.993
ALL	0.994	0.985	0.881	0.547	0.989	0.979
Computed All	0.995	0.994	0.890	0.600	0.991	0.983

Figure 5.6: Precision comparison NSL-KDD.



	DoS	Probe	R2L	U2R	Normal	Weighted avg.
TCP	1.000	0.990	0.960	0.870	0.970	0.980
UDP	0.993	0.988	0.974	1.000	0.969	0.975
OTHER	0.999	0.995	1.000	0.000	0.969	0.993
ALL	0.997	0.993	0.962	0.853	0.954	0.977
Computed All	0.997	0.993	0.958	0.868	0.967	0.982

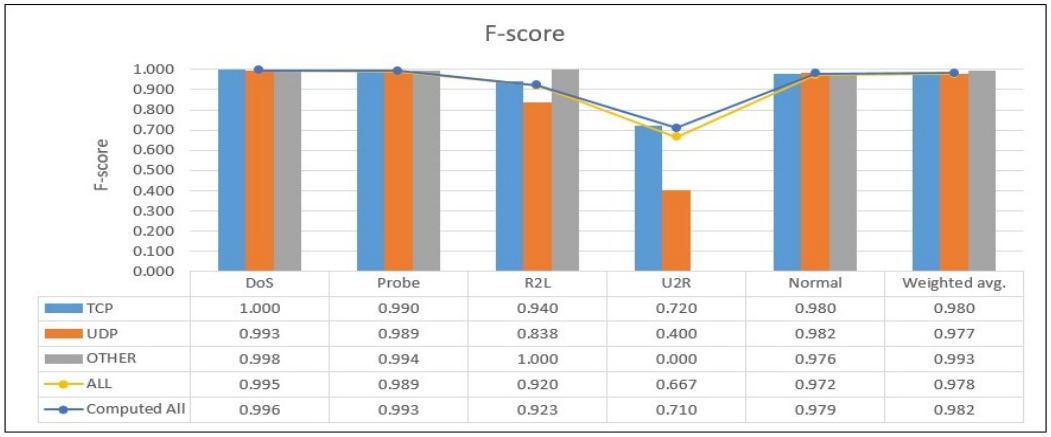
Figure 5.7: Recall comparison NSL-KDD.

Table 5.20: Class-wise Recall comparison with previous works.

Dataset	Class	Vinayakumar Ravi et al. [90]	Yang et al.[77]	Our Method (Computed All)
UNSW-NB15	Analysis	0.00	0.15	**0.59**
	Backdoor	0.34	0.21	**0.66**
	DoS	**0.98**	0.08	0.66
	Exploits	0.01	0.71	**0.76**
	Fuzzers	0.00	0.35	**0.91**
	Generic	0.57	**0.96**	0.79
	Normal	0.93	0.81	**0.99**
	Reconnaissance	0.02	0.80	**0.91**
	Shellcode	0.00	**0.92**	0.73
	Worms	0.00	**0.80**	0.71
NSL-KDD	Normal	0.97	0.97	**0.97**
	DoS	0.80	0.86	**1.00**
	Probe	0.64	0.75	**0.99**
	R2L	0.27	0.44	**0.96**
	U2R	0.23	0.11	**0.87**

5.4.ii. Discussion

To relate the model's performance with the data distribution, overlap problems that exist and to get insight, one way is to visualize the data. There are many visualization techniques like PCA, t-SNE, etc. In [91, 92], the authors used the t-SNE visualization technique to understand the



	DoS	Probe	R2L	U2R	Normal	Weighted avg.
TCP	1.000	0.990	0.940	0.720	0.980	0.980
UDP	0.993	0.989	0.838	0.400	0.982	0.977
OTHER	0.998	0.994	1.000	0.000	0.976	0.993
ALL	0.995	0.989	0.920	0.667	0.972	0.978
Computed All	0.996	0.993	0.923	0.710	0.979	0.982

Figure 5.8: F-score comparison NSL-KDD.

characteristics of datasets. The t-SNE is a non-linear dimensionality reduction technique that turns the high dimensional feature vector into two or more dimensions [90]. We have used t-SNE to visualize protocol-wise data as well as all data. From both figures, we can see that different protocols have different overlap problems. If we treat the data all together without protocol wise separation, it will increase the overlap among all the classes and ultimately will impact the performance of the model. From the comparison of weighted average Precision, Recall, and F-score on both datasets, it is evident that the Computed All performances are higher than the ALL. It means that treating data based on separate protocols results in better performance. There may be two reasons; first, separate protocol data produces different feature sets; second, the overlap problem will be reduced in protocol wise data. Both of these factors improve the performance.

5.5 Conclusion

To understand the characteristics of the datasets used for experiments, we used the t-SNE feature visualization technique to see the overlap problem of data. We observe that both datasets suffer from this issue. Also, the datasets are imbalanced over many classes. We split the data based on different protocols to overcome these problems, and feature selection and classification are conducted on protocol-wise data independently. On the other hand, to solve the imbalance issue, we first applied random undersampling on majority classes and then oversampled the minority classes using the SMOTE technique. One-Dimensional Convolutional Neural Network architecture is used for classification purposes. The model performance is evaluated on the portion of UNSW-NB15 and the portion of NSL-KDD datasets. The experimental results show that treating independent categories yield better results in the case of all metrics we used, such as Precision, Recall and F-score. Also, we compared our results with some of the previous works on overall multiclass classification as well as class-wise classification. Our method outperforms state-of-the-art using NSL-KDD in both types of comparisons. For the UNSW-NB15 dataset, it performs better in most of the classes for class-wise comparison, and for overall comparison, Precision is high from all the compared works, and for Recall and F-score, it is higher than two previous works and lower than one previous work.

Chapter 6

Conclusion and Future Directions

This chapter presents a high-level summary of the research work as detailed in the preceding chapters and incorporates the identified objectives, methods and techniques employed, corresponding experimental results analysis, and discussion. Also, the scientific contribution of our work is enumerated here. In the end, we present some of the identified points for future work. The summarized conclusions of our work are presented in section 6.1, and the future research direction is given in section 6.2.

6.1 Conclusion

In chapter 1, a brief overview of cyber-security and IDS details are provided. The problem statement is presented, and the datasets are described. Sufficient literature, the most recent and most related is reviewed. The review's focus has been on different network intrusion detection datasets, feature selection techniques, class imbalance processing, machine learning-based network anomaly detection, ensemble learning approaches for network anomaly detection, and DL algorithms used in NADS.
Chapter 2 presents an effective feature selection method for network anomaly detection. Five feature selection methods, such as VT, ANOVA, SBS, RFE, and LASSO, are selected from all three feature selection classes: filter method, wrapper method, and embedded method. Each feature selector selects some number of features based on some threshold, and then all the selected sets of features are combined through union and quorum combination methods from set theory. The combined features are evaluated with different classifiers using the cross-validation method. The method iterates the number of features in the sets and plots the metric scores. In this way, the best classifier, the best combination set of features, and the best number of features are selected for network anomaly detection.
Chapter 3 demonstrates the development of network anomaly detection using ensemble learning algorithms consisting of data preprocessing, imbalance processing, and classification. As the data related to network intrusion detection systems are imbalanced in the real world, it affects model performance. We introduced a new imbalance processing method combining the oversampling technique using SMOTE and the cluster-based undersampling technique using the K-means algorithm (SKM). The NAD model fuses SKM and XGBoost, i.e., SKM-XGB. The SKM-XGB outperforms the state-of-the-art methods.
Chapter 4 introduces the development of network anomaly detection using DL algorithms consisting of data preprocessing, imbalance processing, and classification. The data preprocessing in this chapter is similar to data preprocessing in chapter 4. We introduced a new imbalance processing method combining the oversampling technique using ADASYN and a cluster-based undersampling technique using the K-means algorithm. The NAD model fuses imbalance pro-

cessing technology and a 1D CNN-based classifier to detect network anomalies.
In chapter 5, we visualized the data using t-SNE visualization technique based on different protocols. We observed some overlap problems in addition to imbalance issues in both datasets. On the other hand, the overlap problem is different from protocol to protocol. We sampled data using combined resampling techniques consisting of random undersampling and SMOTE to solve the imbalance issue. To mitigate the risk of overlap problems on model performance, we split the data based on protocols and then treat each protocol data independently. Feature selection was conducted independently for each data segment based on protocols. A 1D CNN model was trained and evaluated on separate data, and the final results of the model on different data segments are compared with the results of the model on ALL data. Similarly, the results of the model on ALL data are compared with the calculated results combining the independent segments' results. From these experiments, we observed and proved that treating data based on separate protocols leads to better model performance.

6.2 Open Problems and Future Works

Although the proposed models are performing well in network anomaly detection, some improvements can still be explored. Over the course of this research work, some more questions have been identified than initially proposed. This section, therefore, lists some possible directions for future works.

- Feature selection is a vital part of data preprocessing in network anomaly detection to optimize the computational complexity by identifying a superior set of features. Automating the feature selection process seems to be a promising improvement in the purview of NADS and many other domains since it requires an intensive analysis otherwise. However, it is not an easy task to be carried out because (1) many feature selection techniques rely on a threshold to reduce features. It is a challenging task to define a good termination point or threshold to stop the process of selection. In our experiments, we observed that the testing performance fluctuates over the iterative reduction process even with a single selection method that makes it more challenging to identify a fixed point to stop the process. (2) also, we observed that individual selection methods produce a different subset of features even with the same set of data and the same threshold defined, so it is difficult to decide which set has to be selected as a final set. So, it is a challenging task, but automating the selection process will help NADS to detect anomalies faster without much human interventions.

- Network traffic data is imbalanced in the real world. It causes inefficiency in models' performance. Current resampling techniques partially overcome the issue, not entirely, because oversampling somehow increases data size, and undersampling somehow causes information loss by eliminating samples. We observed that despite resampling and imbalance processing in multiclass classification, the classification is not stable for all the labels with few samples. More work is required to handle the imbalance issue to minimize the risks mentioned above.

- Ensemble learning is one of the solutions for imbalanced data analysis, also there are different ensemble learning paradigms and various combination of ensembling. To select the winner model, an intensive analysis is required. There has to be a profound analysis technique to select the best model for network traffic data in a less comparative analysis to cope with the live traffic in the real world.

- In the real world, the form of attacks and pattern of network traffic may change quickly. Developing an adaptive model to cope with a fast-changing data pattern seems promising.

- The common issues with neural networks are many parameters to be adjusted and the best network architecture to be selected. We observed that changing some parameters with the same network architecture affects the performance; similarly, vice versa is true. Much work is required to find an optimal method to lighten the parameter optimisation process and ease the best architecture selection process.
- Adversarial attacks are techniques of such malicious acts which attempt to fool models with deceptive data and lead to misclassification. However, network packets impose more constraints on their features, leading to lower exposure to adversarial attacks than image data, but the risk is still enormous and requires cautious treatment.
- Overlap problem impacts model performance, especially in the case of multiclass classification. It requires some more work to find ways to treat overlapped classes appropriately.

List of Publications

International Journals

- Mohammad Kazim Hooshmand and Doreswamy. **Machine Learning Based Network Anomaly Detection.** *International journal of recent technology and engineering*, November 2019, ISSN: 2277-3878, Vol. 8, Issue 4, pp. 542–548. https://doi.org/10.35940/ijrte.D7271.118419
- Doreswamy, Mohammad Kazim Hooshmand and Ibrahim Gad. **Feature Selection Approach Using Ensemble Learning for Network Anomaly Detection.** *CAAI Transactions on Intelligence Technology*, October 2020, ISSN: 2468-2322, Vol. 5, Issue 4, pp. 283–293. https://doi.org/10.1049/trit.2020.0073
- Mohammad Kazim Hooshmand and Doreswamy. **Network Anomaly Detection Using Deep Learning Techniques.** *CAAI Transactions on Intelligence Technology*, January 2022, ISSN: 2468-2322, Vol. 7, Issue 2, pp. 228-243. https://doi.org/10.1049/cit2.12078
- Mohammad Kazim Hooshmand and Manjaiah D.H. **Network Intrusion Detection with 1D Convolutional Neural Networks.** *Digital Technologies Research and Applications*, August 2022, ISSN: 2754-5687, Vol. 01, Issue 02, pp. 25-34. https://doi.org/10.54963/dtra.v1i2.64

International Conferences

- Doreswamy and Mohammad Kazim Hooshmand. **Network Traffic Data Classification Using Machine Learning Algorithms.** *International conference on computational intelligence and applications (ICCIA-2019)*, 09 August 2019, pp. 123-128.
- Doreswamy and Mohammad Kazim Hooshmand. **Using Ensemble Learning Approach to Identify Rare Cyber-Attacks in Network Traffic Data**. *IEEE International Conference on Advanced Computer Science and Information Systems (ICACSIS 2020)*, 17 October 2020. pp. 141-146. https://doi.org/10.1109/ICACSIS51025.2020.9263111

Journal Papers Under Preparation

- Mohammad Kazim Hooshmand and Manjaiah D.H. **Network Anomaly Detection Using Ensemble Learning Approach.** Elsevier: Information Fusion.

Bibliography

[1] Faheem Ullah and M Ali Babar. Architectural tactics for big data cybersecurity analytic systems: A review. *arXiv preprint arXiv:1802.03178*, 2018.

[2] Sumeet Dua and Xian Du. *Data mining and machine learning in cybersecurity*. Auerbach Publications, 2016.

[3] Rossouw Von Solms and Johan Van Niekerk. From information security to cyber security. *computers & security*, 38:97–102, 2013.

[4] Salvatore Pontarelli, Giuseppe Bianchi, and Simone Teofili. Traffic-aware design of a high-speed fpga network intrusion detection system. *IEEE Transactions on Computers*, 62(11): 2322–2334, 2012.

[5] Monowar H Bhuyan, Dhruba K Bhattacharyya, and Jugal K Kalita. *Network traffic anomaly detection and prevention: concepts, techniques, and tools*. Springer, 2017.

[6] Elike Hodo, Xavier Bellekens, Andrew Hamilton, Christos Tachtatzis, and Robert Atkinson. Shallow and deep networks intrusion detection system: A taxonomy and survey. *arXiv preprint arXiv:1701.02145*, 2017.

[7] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2): 1153–1176, 2015.

[8] Aleksandar Milenkoski, Marco Vieira, Samuel Kounev, Alberto Avritzer, and Bryan D Payne. Evaluating computer intrusion detection systems: A survey of common practices. *ACM Computing Surveys (CSUR)*, 48(1):1–41, 2015.

[9] Igino Corona, Giorgio Giacinto, and Fabio Roli. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences*, 239:201–225, 2013.

[10] Isra Al-Turaiki and Najwa Altwaijry. A convolutional neural network for improved anomaly-based network intrusion detection. *Big Data*, 9(3):233–252, 2021.

[11] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015.

[12] The UNSW-NB15 data set description, . URL https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets.

[13] The NSL-KDD data set description, . URL https://www.unb.ca/cic/datasets/nsl.html.

[14] Safaa Laqtib, Khalid El Yassini, and Moulay Lahcen Hasnaoui. Evaluation of deep learning approaches for intrusion detection system in manet. In *The Proceedings of the Third International Conference on Smart City Applications*, pages 986–998. Springer, 2019.

[15] OY Al-Jarrah, Amna Siddiqui, M Elsalamouny, Paul D Yoo, Sami Muhaidat, and Kwangjo Kim. Machine-learning-based feature selection techniques for large-scale network intrusion detection. In *2014 IEEE 34th international conference on distributed computing systems workshops (ICDCSW)*, pages 177–181. IEEE, 2014.

[16] Qin Liu, Shihai Shi, Hongming Zhu, and Jiakai Xiao. A mutual information-based hybrid feature selection method for software cost estimation using feature clustering. In *2014 IEEE 38th Annual Computer Software and Applications Conference*, pages 27–32. IEEE, 2014.

[17] Félix Iglesias and Tanja Zseby. Analysis of network traffic features for anomaly detection. *Machine Learning*, 101(1):59–84, 2015.

[18] Ishfaq Manzoor, Neeraj Kumar, et al. A feature reduced intrusion detection system using ann classifier. *Expert Systems with Applications*, 88:249–257, 2017.

[19] Priyanka Dahiya and Devesh Kumar Srivastava. Network intrusion detection in big dataset using spark. *Procedia Computer Science*, 132:253–262, 2018.

[20] Gayatri V Patil, K Vinod Pachghare, and Deepak D Kshirsagar. Feature reduction in flow based intrusion detection system. In *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pages 1356–1362. IEEE, 2018.

[21] Juan Vanerio and Pedro Casas. Ensemble-learning approaches for network security and anomaly detection. In *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, pages 1–6, 2017.

[22] Abdulla Amin Aburomman and Mamun Bin Ibne Reaz. A survey of intrusion detection systems based on ensemble and hybrid classifiers. *Computers & Security*, 65:135–152, 2017.

[23] Nour Moustafa, Benjamin Turnbull, and Kim-Kwang Raymond Choo. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet of Things Journal*, 6(3):4815–4830, 2018.

[24] Marzia Zaman and Chung-Horng Lung. Evaluation of machine learning techniques for network intrusion detection. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–5. IEEE, 2018.

[25] Xianwei Gao, Chun Shan, Changzhen Hu, Zequn Niu, and Zhen Liu. An adaptive ensemble machine learning model for intrusion detection. *IEEE Access*, 7:82512–82521, 2019.

[26] Han Liu, Xianchao Zhang, and Xiaotong Zhang. Pwadaboost: Possible world based adaboost algorithm for classifying uncertain data. *Knowledge-Based Systems*, 186:104930, 2019.

[27] Saurabh Sharma, Ning Yu, Mario Fritz, and Bernt Schiele. Long-tailed recognition using class-balanced experts. In *DAGM German Conference on Pattern Recognition*, pages 86–100. Springer, 2020.

[28] Dan Tang, Liu Tang, Rui Dai, Jingwen Chen, Xiong Li, and Joel JPC Rodrigues. Mf-adaboost: Ldos attack detection based on multi-features and improved adaboost. *Future Generation Computer Systems*, 106:347–359, 2020.

[29] Ivo Berger, Roland Rieke, Maxim Kolomeets, Andrey Chechulin, and Igor Kotenko. Comparative study of machine learning methods for in-vehicle intrusion detection. In *Computer Security*, pages 85–101. Springer, 2018.

[30] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access*, 5:21954–21961, 2017.

[31] Dominik Heider, Christoph Bartenhagen, J Nikolaj Dybowski, Sascha Hauke, Martin Pyka, and Daniel Hoffmann. Unsupervised dimension reduction methods for protein sequence classification. In *Data analysis, machine learning and knowledge discovery*, pages 295–302. Springer, 2014.

[32] Mohamed Amine Ferrag, Leandros Maglaras, Sotiris Moschoyiannis, and Helge Janicke. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50:102419, 2020.

[33] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. A deep learning approach to network intrusion detection. *IEEE transactions on emerging topics in computational intelligence*, 2(1):41–50, 2018.

[34] Arwa Aldweesh, Abdelouahid Derhab, and Ahmed Z Emam. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, 189:105124, 2020.

[35] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. Applying convolutional neural network for network intrusion detection. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1222–1228. IEEE, 2017.

[36] Kehe Wu, Zuge Chen, and Wei Li. A novel intrusion detection model for a massive network using convolutional neural networks. *Ieee Access*, 6:50850–50859, 2018.

[37] Nathalie Japkowicz. The class imbalance problem: Significance and strategies. In *Proc. of the Int'l Conf. on Artificial Intelligence*, volume 56, pages 111–117. Citeseer, 2000.

[38] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2011.

[39] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE, 2008.

[40] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16: 321–357, 2002.

[41] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2008.

[42] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[43] David A Cieslak, Nitesh V Chawla, and Aaron Striegel. Combating imbalance in network intrusion datasets. In *GrC*, pages 732–737. Citeseer, 2006.

[44] Pandit Byomakesha Dash, Janmenjoy Nayak, Bighnaraj Naik, Etuari Oram, and SK Hafizul Islam. Model based iot security framework using multiclass adaptive boosting with smote. *Security and Privacy*, 3(5):e112, 2020.

[45] Wenming Huan, Haitao Lin, Haixue Li, Yan Zhou, and Yiming Wang. Anomaly detection method based on clustering undersampling and ensemble learning. In *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pages 980–984. IEEE, 2020.

[46] Abhishek Divekar, Meet Parekh, Vaibhav Savla, Rudra Mishra, and Mahesh Shirole. Benchmarking datasets for anomaly-based network intrusion detection: Kdd cup 99 alternatives. *arXiv preprint arXiv:1811.05372*, 2018.

[47] Wajdi Alhakami, Abdullah ALharbi, Sami Bourouis, Roobaea Alroobaea, and Nizar Bouguila. Network anomaly intrusion detection using a nonparametric bayesian approach and feature selection. *IEEE Access*, 7:52181–52190, 2019.

[48] Yali Yuan, Georgios Kaklamanos, and Dieter Hogrefe. A novel semi-supervised adaboost technique for network anomaly detection. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 111–114, 2016.

[49] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. *Feature selection for high-dimensional data*. Springer, 2015.

[50] Salem Alelyani, Jiliang Tang, and Huan Liu. Feature selection for clustering: A review. In *Data Clustering*, pages 29–60. Chapman and Hall/CRC, 2018.

[51] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–45, 2017.

[52] Guansong Pang, Longbing Cao, and Ling Chen. Outlier detection in complex categorical data by modelling the feature value couplings. In *IJCAI International Joint Conference on Artificial Intelligence*, 2016.

[53] Guansong Pang, Longbing Cao, Ling Chen, and Huan Liu. Learning homophily couplings from non-iid data for joint feature selection and noise-resilient outlier detection. In *IJCAI*, pages 2585–2591, 2017.

[54] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. Network anomaly detection: methods, systems and tools. *Ieee communications surveys & tutorials*, 16(1):303–336, 2013.

[55] Abien Fred M Agarap. A neural network architecture combining gated recurrent unit (gru) and support vector machine (svm) for intrusion detection in network traffic data. In *Proceedings of the 2018 10th international conference on machine learning and computing*, pages 26–30, 2018.

[56] Wei Zong, Yang-Wai Chow, and Willy Susilo. Interactive three-dimensional visualization of network intrusion detection data for machine learning. *Future Generation Computer Systems*, 102:292–306, 2020.

[57] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Jill Slay. Towards developing network forensic mechanism for botnet activities in the iot based on machine learning techniques. In *International Conference on Mobile Networks and Management*, pages 30–44. Springer, 2017.

[58] T Tulasi Bhavani, M Kameswara Rao, and A Manohar Reddy. Network intrusion detection system using random forest and decision tree machine learning techniques. In *First international conference on sustainable technologies for computational intelligence*, pages 637–643. Springer, 2020.

[59] Phurivit Sangkatsanee, Naruemon Wattanapongsakorn, and Chalermpol Charnsripinyo. Practical real-time intrusion detection using machine learning approaches. *Computer Communications*, 34(18):2227–2235, 2011.

[60] Adel Binbusayyis and Thavavel Vaiyapuri. Identifying and benchmarking key features for cyber intrusion detection: An ensemble approach. *IEEE Access*, 7:106495–106513, 2019.

[61] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[62] Abdulla Amin Aburomman and Mamun Bin Ibne Reaz. A novel svm-knn-pso ensemble method for intrusion detection system. *Applied Soft Computing*, 38:360–372, 2016.

[63] Hongpo Zhang, Chase Q Wu, Shan Gao, Zongmin Wang, Yuxiao Xu, and Yongpeng Liu. An effective deep learning based scheme for network intrusion detection. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 682–687. IEEE, 2018.

[64] Nour Moustafa and Jill Slay. Rcnf: Real-time collaborative network forensic scheme for evidence analysis. *arXiv preprint arXiv:1711.02824*, 2017.

[65] Jingmei Liu, Yuanbo Gao, and Fengjie Hu. A fast network intrusion detection system using adaptive synthetic oversampling and lightgbm. *Computers & Security*, 106:102289, 2021.

[66] Mustapha Belouch, Salah El Hadaj, and Mohamed Idhammad. A two-stage classifier approach using reptree algorithm for network intrusion detection. *International Journal of Advanced Computer Science and Applications*, 8(6):389–394, 2017.

[67] Hamid Tahaei, Firdaus Afifi, Adeleh Asemi, Faiz Zaki, and Nor Badrul Anuar. The rise of traffic classification in iot networks: A survey. *Journal of Network and Computer Applications*, 154:102538, 2020.

[68] Mario Luca Bernardi, Marta Cimitile, Damiano Distante, Fabio Martinelli, and Francesco Mercaldo. Dynamic malware detection and phylogeny analysis using process mining. *International Journal of Information Security*, 18(3):257–284, 2019.

[69] Giovanni Acampora, Mario Luca Bernardi, Marta Cimitile, Genoveffa Tortora, and Autilia Vitiello. A fuzzy clustering-based approach to study malware phylogeny. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2018.

[70] Chao Liu, Zhaojun Gu, and Jialiang Wang. A hybrid intrusion detection system based on scalable k-means+ random forest and deep learning. *IEEE Access*, 9:75729–75740, 2021.

[71] Khalil El-Khatib. Impact of feature reduction on the efficiency of wireless intrusion detection systems. *IEEE TRANSACTIONS on parallel and distributed systems*, 21(8):1143–1149, 2009.

[72] Hao Zhang, Jie-Ling Li, Xi-Meng Liu, and Chen Dong. Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection. *Future Generation Computer Systems*, 122:130–143, 2021.

[73] Gulshan Kumar. An improved ensemble approach for effective intrusion detection. *The Journal of Supercomputing*, 76(1):275–291, 2020.

[74] Sanjit Maitra, Rahul Kumar Ojha, and Kuntal Ghosh. Impact of convolutional neural network input parameters on classification performance. In *2018 4th International Conference for Convergence in Technology (I2CT)*, pages 1–5. IEEE, 2018.

[75] Long Wen, Liang Gao, Xinyu Li, and Bing Zeng. Convolutional neural network with automatic learning rate scheduler for fault classification. *IEEE Transactions on Instrumentation and Measurement*, 70:1–12, 2021.

[76] Mirza M Baig, Mian M Awais, and El-Sayed M El-Alfy. A multiclass cascade of artificial neural network for network intrusion detection. *Journal of Intelligent & Fuzzy Systems*, 32 (4):2875–2883, 2017.

[77] Yanqing Yang, Kangfeng Zheng, Chunhua Wu, and Yixian Yang. Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. *Sensors*, 19(11):2528, 2019.

[78] Tao Ma, Fen Wang, Jianjun Cheng, Yang Yu, and Xiaoyun Chen. A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. *Sensors*, 16(10):1701, 2016.

[79] Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot. *Sensors*, 17(9):1967, 2017.

[80] Yadigar Imamverdiyev and Fargana Abdullayeva. Deep learning method for denial of service attack detection based on restricted boltzmann machine. *Big data*, 6(2):159–169, 2018.

[81] Zeinab Zoghi and Gursel Serpen. Unsw-nb15 computer security dataset: Analysis through visualization. *arXiv preprint arXiv:2101.05067*, 2021.

[82] Vaishali Ganganwar. An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, 2(4):42–47, 2012.

[83] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

[84] Enislay Ramentol, Yailé Caballero, Rafael Bello, and Francisco Herrera. Smote-rs b*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory. *Knowledge and information systems*, 33(2):245–265, 2012.

[85] Alexander Liu, Joydeep Ghosh, and Cheryl E Martin. Generative oversampling for mining imbalanced datasets. In *DMIN*, pages 66–72, 2007.

[86] Chetna Kumari, Muhammad Abulaish, and Naidu Subbarao. Using smote to deal with class-imbalance problem in bioactivity data to predict mtor inhibitors. *SN Computer Science*, 1:1–7, 2020.

[87] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[88] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[89] S Sriram, A Shashank, R Vinayakumar, and KP Soman. Dcnn-ids: Deep convolutional neural network based intrusion detection system. In *International Conference on Computational Intelligence, Cyber Security, and Computational Models*, pages 85–92. Springer, 2019.

[90] Ravi Vinayakumar, Mamoun Alazab, KP Soman, Prabaharan Poornachandran, Ameer Al-Nemrat, and Sitalakshmi Venkatraman. Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 7:41525–41550, 2019.

[91] S Sriram, R Vinayakumar, Mamoun Alazab, and KP Soman. Network flow based iot botnet attack detection using deep learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 189–194. IEEE, 2020.

[92] R Vinayakumar, Mamoun Alazab, Sriram Srinivasan, Quoc-Viet Pham, Soman Kotti Padannayil, and K Simran. A visualized botnet detection system based deep learning for the internet of things networks of smart cities. *IEEE Transactions on Industry Applications*, 56(4):4436–4456, 2020.

Research candidate	**Research supervisor**
Mohammad Kazim **Research scholar,** **Dept. of Computer Science,** **Mangalore University.**	**Dr. Manjaiah D.H,** **Professor,** **Dept. of Computer Science,** **Mangalore University.**